L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
_$
Ps
--
YZ
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
```

```
**FILE**ID**LNMSUB
```

```
LL          NN      NN  MM      MM    SSSSSSSS  UU      UU  BBBBBBBB
LL          NN      NN  MM      MM    SSSSSSSS  UU      UU  BBBBBBBB
LL          NN      NN  MMMM  MMMM  SS          UU      UU  BB      BB
LL          NN      NN  MMMM  MMMM  SS          UU      UU  BB      BB
LL          NNNN    NN  MM  MM  MM  SS          UU      UU  BB      BB
LL          NNNN    NN  MM  MM  MM  SS          UU      UU  BB      BB
LL          NN  NN  NN  MM      MM    SSSSSS    UU      UU  BBBBBBBB
LL          NN  NN  NN  MM      MM    SSSSSS    UU      UU  BBBBBBBB
LL          NN    NNNN  MM      MM          SS  UU      UU  BB      BB
LL          NN    NNNN  MM      MM          SS  UU      UU  BB      BB
LL          NN      NN  MM      MM          SS  UU      UU  BB      BB      ....
LL          NN      NN  MM      MM          SS  UU      UU  BB      BB      ....
LLLLLLLLLL  NN      NN  MM      MM  SSSSSSSS    UUUUUUUUUU  BBBBBBBB        ....
LLLLLLLLLL  NN      NN  MM      MM  SSSSSSSS    UUUUUUUUUU  BBBBBBBB        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000            .TITLE  LNMSUB - LOGICAL NAME RELATED SUBROUTINES
0000            .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ; LOGICAL NAME RELATED SUBROUTINES ( REPLACES LOGNAMSUB MODULE )
0000    ;
0000    ; David W. Thiel    29-Oct-1982
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-031 RAS0330         Ron Schaefer            31-Jul-1984
0000    ;               Fix basic name lookup algorithm so that case blind
0000    ;               lookups can find lower case names.
0000    ;
0000    ;       V03-030 ACG0440         Andrew C. Goldstein,    24-Jul-1984  10:36
0000    ;               Add ref count field to ORB
0000    ;
0000    ;       V03-029 LMP0275         L. Mark Pilant,         12-Jul-1984  20:03
0000    ;               Initialize the ACL info in the ORB to be a null descriptor
0000    ;               list rather than an empty queue.  This avoids the overhead
0000    ;               of locking and unlocking the ACL mutex, only to find out
0000    ;               that the ACL was empty.
0000    ;
0000    ;       V03-028 RAS0319         Ron Schaefer                    29-Jun-1984
0000    ;               Add a simple logical name table name translation cache
0000    ;               to the recursive table name lookup algorithm in order to
0000    ;               (hopefully) improve the performance of logical name processing.
0000    ;               The methodology is as follows:
0000    ;
0000    ;               In per-process space, there exists a small queue of fixed-len
0000    ;               entries that contain a table name LNMB addr, a process and
0000    ;               system directory incarnation sequence number and upto
0000    ;               LNMC$K_NUM_ENTRIES worth of table header (LNMTH) addresses
0000    ;               that are the recursive tables identified by that table name.
```

```
0000  ;                When doing a table lookup;
0000  ;                We lookup the table name
0000  ;                normally.  We then check the cache for a matching LNMB.
0000  ;                If there is a cache entry and the directory incarnation numbers
0000  ;                are still valid, then we use the vector of LNMTH entries
0000  ;                rather than looking them up by name.
0000  ;                If no valid cache entry is found, we take the LRU cache
0000  ;                entry, initialize it for this name and use the normal
0000  ;                table lookup procedures, except that as a table header is
0000  ;                found, we store it's address in the cache entry for future
0000  ;                reference.
0000  ;                If we use up all cache entries before finding the correct
0000  ;                table, the cache entries will run out.  In that case, we
0000  ;                reset the cache index to start building a new cache entry
0000  ;                table from the beginning since it is impossible to relate
0000  ;                a given cache entry back to the corresponding name.
0000  ;                If the number of table entries, exceeds the cache table size
0000  ;                then we give up and do things the slow way.
0000  ;
0000  ;        V03-027 RAS0316         Ron Schaefer                    25-Jun-1984
0000  ;                Add new LNM$DELETE_HASH routine to delete all logical names
0000  ;                within a given hash table and having a given or outer
0000  ;                access mode.  This is ECO 4 in the FT2 update.
0000  ;
0000  ;        V03-026 RAS0312         Ron Schaefer                    18-Jun-1984
0000  ;                Prevent accvio in searching tables if no process logical
0000  ;                name table/hash table/directory has been created.
0000  ;                Fix alignment of ORB block in table.
0000  ;
0000  ;        V03-025 TMK0018         Todd M. Katz                    24-Apr-1984
0000  ;                Change the name of the logical name mutex from LOG$AL_MUTEX
0000  ;                to LNM$AL_MUTEX.
0000  ;
0000  ;        V03-024 TMK0017         Todd M. Katz                    21-Apr-1984
0000  ;                I have changed the interface to the routine LNM$DELETE_LNMB.
0000  ;                This routine is now called with the logical name block it is
0000  ;                to delete together with all of the outer access mode aliases
0000  ;                of the logical name represented by the logical name block.
0000  ;                Previously the interface to this routine consisted of it being
0000  ;                called with a logical name descriptor, containing table header
0000  ;                address, and the access mode of the innermost logical name
0000  ;                to be deleted, despite the fact that the calling routine always
0000  ;                had the logical name block for the innermost access mode logical
0000  ;                name to be deleted.
0000  ;
0000  ;                Make LNM$DELETE a local routine. This routine is only called by
0000  ;                the routines within this module.
0000  ;
0000  ;                The performance measurement cell used to monitor the rate of
0000  ;                logical name translations is currently located within the
0000  ;                internal logical name routine LNM$SEARCHLOG. Unfortunately,
0000  ;                because of its current placement, any attempts to delete
0000  ;                specific logical names will also increment this counter. This
0000  ;                is because the system service $DELLNM will call the routine
0000  ;                LNM$SEARCHLOG in such a situation. Therefore, in order to be
0000  ;                able to make a more accurate measurement of the overall rate
0000  ;                of logical name translations, I have decided to move this
```

```
0000  ;                 performance measurement cell from its current single location
0000  ;                 to several more appropriate locations. One of these new
0000  ;                 locations is within the internal routine LNM$SEARCH_ONE just
0000  ;                 before the call to LNM$SEARCHLOG.
0000  ;
0000  ;         V03-023 CWH3023         CW Hobbs                        14-Apr-1984
0000  ;                 Fix a broken branch.
0000  ;
0000  ;         V03-022 TMK0016         Todd M. Katz                    11-Apr-1984
0000  ;                 Make a change to DELETE_LNMB such that when this routine is
0000  ;                 called to delete a shareable logical name table and its
0000  ;                 associated Object Rights Block, it cleanups up the ORB first
0000  ;                 before deleting the logical name block and ORB.
0000  ;
0000  ;         V03-021 TMK0015         Todd M. Katz                    10-Apr-1984
0000  ;                 Change LNM$SEARCH_ONE to position past all translation blocks
0000  ;                 with negative indexes (ie - those reserved for system use) to
0000  ;                 the first translation block with a non-negative index or to the
0000  ;                 last translation block, whichever comes first. If the
0000  ;                 translation block positioned to is not the last one and has an
0000  ;                 index of 0, then return success together with the contents of
0000  ;                 the entire translation block in the user supplied buffer.
0000  ;                 Otherwise, return an error of SS$_NOLOGNAM.
0000  ;
0000  ;         V03-020 TMK0014         Todd M. Katz                    02-Apr-1984
0000  ;                 I have made the following optimizations and bug fixes to the
0000  ;                 routines within this module.
0000  ;
0000  ;                 1. I have changed how LNMBs are ordered within a hash bucket
0000  ;                    both to increase performance and to fix a problem in hash
0000  ;                    bucket searching. The new ordering is first by name string
0000  ;                    length, then my name string, then by containing table header
0000  ;                    address, and finally by access mode. Previously, the LNMBs
0000  ;                    were ordered by access mode before containing table header
0000  ;                    address.
0000  ;
0000  ;                    What this new ordering gains is the ability to use the
0000  ;                    containing table header address in determining when searches,
0000  ;                    such as those directed by LNM$SEARCHLOG, should terminate for
0000  ;                    a current containing table header address. With the old
0000  ;                    ordering, containing table header addresses could not be used
0000  ;                    to terminate a search. Even if the target LNMB was not found
0000  ;                    among the user mode LNMBs, the search would still have to
0000  ;                    continue with the supervisor, executive, and kernel mode
0000  ;                    LNMBs. Basically, this meant ignoring the ordering of LNMBs
0000  ;                    by containing table header addresses. Unfortunately, this
0000  ;                    "ignoring" was not being done, and this resulted in the
0000  ;                    inability to find certain logical names even though they were
0000  ;                    present. When I changed the ordering of LNMBs within the hash
0000  ;                    bucket, not only did I gain the ability to use the ordering
0000  ;                    of LNMBs by containing table header addresses to terminate a
0000  ;                    search, and thus increase performance,  but at the same time
0000  ;                    I eliminated this particular problem which was causing
0000  ;                    certain $TRNLNMs to fail when they should have succeeded.
0000  ;
0000  ;                 2. The routine LNM$CONTSEARCH makes the assumption that when
0000  ;                    it is directly called, NT_L_THREAD contains the address
```

```
0000   ;            of the previous LNMB block, and the search is to continue
0000   ;            with the LNMB which follows it. If there is no possibility
0000   ;            that the contents of the hash bucket could have changed since
0000   ;            the last search was done, then the name string of the first
0000   ;            LNMB this routine looks at, the one pointed at by the LNMB
0000   ;            whose address is contained within NL$L_THREAD, is guarenteed
0000   ;            to match the target name string. Therefore, there is no need
0000   ;            to make the string comparison which is normally done because
0000   ;            the outcome is already known, and in fact, an increase in
0000   ;            performance can be realized if the execution of this CMPC3
0000   ;            is eliminated.
0000   ;
0000   ;            What I have done is define a bit NT_V_MODIFY within the
0000   ;            NT_B_FLAGS field of the translation block. This bit is
0000   ;            set as part of the initialization of a name translation
0000   ;            block whenever the possibility exists that LNM$CONTSEARCH
0000   ;            maybe called after modifying the sequence of LNMBs residing
0000   ;            within a hash bucket. At the present two routines,
0000   ;            LNM$INSLOGTAB and LNM$DELETE_LNMB, set this bit as part of
0000   ;            the process of allocating and initializing their name
0000   ;            translation blocks. When LNM$CONTSEARCH notices that this bit
0000   ;            is clear it skips the initial CMPC3 assuming that the name
0000   ;            strings are equivalent. Only this initial CMPC3 can be
0000   ;            skipped - all remaining string comparisons are required and
0000   ;            are performed as is found to be necesary.
0000   ;
0000   ;         3. I have also made numerous micro-optimizations to the routine
0000   ;            within this module.
0000   ;
0000   ;         4. I have changed this module to use the symbol LNM$C_MAXDEPTH
0000   ;            to define the maximum logical name recursion depth instead
0000   ;            of the local symbol RT_C_MAXDEP.
0000   ;
0000   ;         5. I have made two changes to LNM$SEARCH_ONE. Before calling
0000   ;            LNM$LOCKR to lock the logical name mutex for reading, this
0000   ;            routine saves the current IPL on the stack, and then restores
0000   ;            IPL to this value after unlocking the logical name mutex by
0000   ;            calling LNM$UNLOCK. This change is required because LNM$LOCKR
0000   ;            exits with IPL set to IPL$_AST, and therefore, IPL may have
0000   ;            to be set back to its inital value before LNM$SEARCH_ONE
0000   ;            exits. The reason why IPL is not lowered until after
0000   ;            LNM$UNLOCK returns is that the mutex locking and unlocking
0000   ;            routines make the assumption that IPL does not fall below
0000   ;            IPL$_AST while the current process has the mutex locked.
0000   ;
0000   ;            The second change I have made is a bug fix. LNM$SEARCH_ONE
0000   ;            was checking access to process-private logical name tables
0000   ;            when the logical name found was contained within one of them.
0000   ;            Such a check is un-necessary, and in fact must not be done.
0000   ;            Such tables do not have an Object Rights Block associated
0000   ;            with them, and LNM$CHECK_PROT will accvio if it is called
0000   ;            with a table header that does not have an associated ORB.
0000   ;
0000   ;         6. I have fixed a day-one implementation problem in LNM$HASH
0000   ;            that was making the hashing routine sensitive to the
0000   ;            alignment of the address of the string this routine is
0000   ;            supposed to hash (and why this was ever working I'll never
0000    ;               know). What LNM$HASH should have been doing, and does now is:
0000    ;
0000    ;               1.) Save the number of bytes in the string.
0000    ;               2.) Divide the number of bytes in the string by 4.
0000    ;               3.) Process the string, four bytes at a time, to compute the
0000    ;                   hash code.
0000    ;               4.) Retrieve the string size, and use the low-order two bits,
0000    ;                   in effect the remainder from the integer division in 1.),
0000    ;                   to determine how many bytes of the string have not yet
0000    ;                   participated in the computation of the hash code, and to
0000    ;                   direct their participation.
0000    ;               5.) Complete computation of the hash code.
0000    ;
0000    ;               In step 1.), instead of saving the number of bytes in the
0000    ;               string, LNM$HASH was saving the string address. This made
0000    ;               the computation of the hash code sensitive to the alignment
0000    ;               of the string address passed to it. Inotherwards, if the
0000    ;               name string FOO was presented to LNM$HASH as being at address
0000    ;               500 at one time, and as being at address 601 a second time,
0000    ;               the two hash code values determined for FOO would be
0000    ;               different - an extremely serious problem. A given name must
0000    ;               always hash to a constant value.
0000    ;
0000    ;       V03-019 TMK0013         Todd M. Katz                    30-Mar-1984
0000    ;               Modify the logical name system services to make use of the
0000    ;               updated internal protection checking mechanisms. What this
0000    ;               requires is modification to the routine LNM$INIT_PROT, so that
0000    ;               it initializes a quad-word aligned Object Rights Block for
0000    ;               shareable logical name tables in place of an un-aligned CHIP
0000    ;               protection template, and a modification to the routine
0000    ;               LNM$CHECK_PROT, so that it makes use of the new internal
0000    ;               check protection system service interface.
0000    ;
0000    ;       V03-018 TMK0012         Todd M. Katz                    22-Mar-1984
0000    ;               Fix a bug in logical name table processing. When a translation
0000    ;               has the attribute TERMINAL, the translation string must be the
0000    ;               the name of a logical name table, and LNM$TABLE remembers this
0000    ;               for the next level of recursion by setting the bit RT_V_TERM
0000    ;               within the RT_B_FLAGS field of the table recursion control
0000    ;               block. If the translation string is the name of a logical name
0000    ;               table then LNM$TABLE makes a successful exit with this bit
0000    ;               still set within the recursion control block. Then if for
0000    ;               some reason LNM$TABLE is called once more to continue the
0000    ;               recursion and find the next table, because RT_V_TERM has not
0000    ;               been cleared, LNM$TABLE will immediately and incorrectly exit
0000    ;               with an error of SS$_IVLOGTAB. The fix to this problem is to
0000    ;               unconditionally clear this bit on entry to LNM$TABLE.
0000    ;
0000    ;       V03-017 TMK0011         Todd M. Katz                    21-Mar-1984
0000    ;               Make LNM$PRESEARCH, LNM$CONTSEARCH, LNM$SETUP, and LNM$TABLE
0000    ;               global, so that they maybe used by SHOW LOGICAL.
0000    ;
0000    ;       V03-016 TMK0010         Todd M. Katz                    07-Mar-1984
0000    ;               Logical name table name processing is recursive. A table
0000    ;               name is provided, and the corresponding logical name block is
0000    ;               located, if one exists, by hashing the name and looking for the
0000    ;               logical name block within the indicated hash buckets (first
0000   ;       searching the process-private name space and then searching the
0000   ;       system name space). If the logical name block exists, and it is
0000   ;       for a logical name table, then table processing terminates.
0000   ;       Otherwise, this procedure is repeated, in turn, for each of the
0000   ;       logical name's translations and for each of the translation's
0000   ;       translations, etc... until the first logical name table is
0000   ;       located, or until all possible translation paths have been
0000   ;       exhausted.
0000   ;
0000   ;       I have added an optimization to this recursive logical name
0000   ;       table name processing. This optimization consists of storing
0000   ;       the hash code value of each equivalence string within the
0000   ;       corresponding translation block. Because the very first step in
0000   ;       each recursive logical name block lookup is the hashing of the
0000   ;       target block's name string in order to provide the hash buckets
0000   ;       in which to concentrate the search, already having the
0000   ;       appropriate hash code means that this step of a logical name
0000   ;       block lookup maybe by-passed.
0000   ;
0000   ;       There are two exceptions to this optimization which will require
0000   ;       the target block's name to be hashed during a recursive logical
0000   ;       name block lookup. First, the target block's name will have to
0000   ;       be hashed during the very first recursive logical name block
0000   ;       lookup. This is because the hash value of this name string isn't
0000   ;       available. Secondly, because there is no way to distinquish
0000   ;       between a valid hash code of 0, and the total absence of a hash
0000   ;       code, this means that whenever one of these stored hash code
0000   ;       values is 0, the equivalence string within the corresponding
0000   ;       translation block will end up being hashed anyway during the
0000   ;       lookup of the target logical name block.
0000   ;
0000   ;       The modifications which are required in order to implement this
0000   ;       feature are as follows:
0000   ;
0000   ;       1. Increase the size of the fixed portion of each translation
0000   ;          block by a word in both logical names and logical name
0000   ;          tables. This word may potentially contain the translation
0000   ;          string's hash code value.
0000   ;
0000   ;       2. Initialize this new field of each translation block with the
0000   ;          hash code value of the corresponding equivalence string
0000   ;          provided the translation block is part of a logical name
0000   ;          (and not a logical name table) contained within a directory
0000   ;          table. It is these names, and only these names, which are
0000   ;          utilized in logical name table name processing, and thus,
0000   ;          only these names have to have the hash code values of their
0000   ;          equivalence strings computed and stored appropriately.
0000   ;
0000   ;          The routine LNM$INSLOGTAB has been modified to perform this
0000   ;          initialization whenever it determines that the logical name
0000   ;          block it is inserting is for a logical name contained within
0000   ;          the process or system directory logical name table.
0000   ;
0000   ;       3. The routine LNM$LOOKUP has been modified, so that it takes
0000   ;          as additional input the hash code value of name string of the
0000   ;          target logical name block this routine is to look up. This
0000   ;          hash code value is placed into the hash function field of a
```

```
0000  ;                         name translation control block. This will result in the
0000  ;                         by-passing of the determination of the hash code value of the
0000  ;                         name string whenever the inputed hash code value is non-zero.
0000  ;
0000  ;                      4. The routine LNM$TABLE has been modified, so that the hash
0000  ;                         code value of the name string of the next target logical name
0000  ;                         block is extracted from the translation block containing the
0000  ;                         name string. This hash code value, together with the name
0000  ;                         string, is then passed to LNM$LOOKUP forcing by-passing of
0000  ;                         the determination of the hash code value of the name string
0000  ;                         of the next target logical name block during its lookup.
0000  ;
0000  ;                      5. Finally, a modification has been made to LNM$SETUP, the
0000  ;                         routine responsible for setting up and initialization the
0000  ;                         recursive logical name table name processing. Because the
0000  ;                         hash code of the name string of the initial target logical
0000  ;                         name block is not available, this routine must zero out the
0000  ;                         register in which LNM$LOOKUP expects to find the hash code
0000  ;                         of the logical name block it is to lookup. This will force
0000  ;                         the hash code value of this name string to be determined,
0000  ;                         which is proper since LNM$SETUP doesn't have it anyway.
0000  ;
0000  ;       V03-015 RAS0255         Ron Schaefer                    17-Feb-1984
0000  ;               Make LNM$M_CASE_BLIND work for DEC multinational characters.
0000  ;
0000  ;       V03-014 TMK0009         Todd M. Katz                    03-Feb-1984
0000  ;               If LNM$FIRSTTAB is unable to find an existing logical name
0000  ;               table within the input list of logical name tables names,
0000  ;               return an error of SS$_NOLOGTAB instead of an error of
0000  ;               SS$_NOLOGNAM.
0000  ;
0000  ;       V03-013 TMK0008         Todd M. Katz                    29-Dec-1983
0000  ;               Add the global routine LNM$DELETE_LNMB which takes as input
0000  ;               the address of a table header, a descriptor of the name of a
0000  ;               logical name table entry, and an access mode and deletes all
0000  ;               instances of the logical name table entry within the specified
0000  ;               logical name table at all access modes outer and equal to that
0000  ;               of the specified access mode.
0000  ;
0000  ;               Add the global routine LNM$SEARCH_ONE which takes as input:
0000  ;
0000  ;               1. A descriptor of a logical name table name.
0000  ;               2. A descriptor of a logical name.
0000  ;               3. A descriptor of an output buffer.
0000  ;               4. The PCB address.
0000  ;               5. An access mode.
0000  ;
0000  ;               This routine simulates a $TRNLNM returning a copy of the LNMX
0000  ;               translation block for translation index 0 in the specified
0000  ;               output buffer provided a logical name is found, and the found
0000  ;               logical name has a translation with an index of 0. This routine
0000  ;               does NOT do any type of arguement verification.
0000  ;
0000  ;               Make a small modification to DELETE_NAMES and LNM$DELETE
0000  ;               so that the directories, process and system, can never be
0000  ;               either explicitely or implicitely deleted. Also, return an
0000  ;               error from within LNM$INSLOGTAB if the caller attempted to
```

```
0000   ;           create a logical name table entry within one of the directories
0000   ;           with the same name and access mode as the directory itself.
0000   ;           This is done by never allowing a LNMB which has the
0000   ;           LNMB$V_NODELETE bit set within its LNMB$B_FLAGS field from being
0000   ;           deleted, and as the directory tables are the only tables that
0000   ;           are created with this attribute, this protects the directory
0000   ;           tables, and only the directory tables, from being deleted.
0000   ;
0000   ;           Make the global routines LNM$PRESEARCH, LNM$CONTSEARCH,
0000   ;           LNM$LOOKUP, LNM$SETUP, AND LNM$TABLE local routines.
0000   ;
0000   ;   V03-012 TMK0007         Todd M. Katz                    27-Dec-1983
0000   ;           LNM$INIT_PROT was clearing the member field of the CHIP template
0000   ;           UIC if the logical name table whose CHIP template was being
0000   ;           inited was a group table. This is no longer necessary because
0000   ;           group tables are now handcrafted, and can not be created by
0000   ;           means of the $CRELNT system service.
0000   ;
0000   ;   V03-011 TMK0006         Todd M. Katz                    14-Dec-1983
0000   ;           Make a change to recursive table lookup processing implemented
0000   ;           within the routine LNM$TABLE. Currently when the lookup of a
0000   ;           name fails it also terminates the recursive search for a logical
0000   ;           name table. The change is to not have a name lookup failure
0000   ;           automatically terminate a table lookup. Instead when the lookup
0000   ;           of a translation string fails, processing continues with the
0000   ;           next translation and an attempt is made to lookup its string.
0000   ;           Conceptually what this means is that non-existant logical name
0000   ;           tables in a list of logical name table names are "skipped over"
0000   ;           during the search for valid logical name tables. Currently, what
0000   ;           would happen is that such a search for valid tables will
0000   ;           immediately terminate with an error when the first non-existant
0000   ;           table was encountered in the list of logical name table names.
0000   ;
0000   ;           Also, change some PUSHRs into PUSHLs (or MOVQs) and POPRs into
0000   ;           POPLs (or MOVQs) where appropriate for performance reasons.
0000   ;
0000   ;   V03-010 TMK0005         Todd M. Katz                    26-Oct-1983
0000   ;           Quota handling when logical name tables are being created or
0000   ;           deleted is presently incorrect. Currently, when a logical name
0000   ;           table is created, quota consisting of the size of the new
0000   ;           logical name table plus any quota explicitely allocated to the
0000   ;           new table is subtracted from the quota holder of the parent
0000   ;           logical name. When a logical name table is deleted, the reverse
0000   ;           of this quota deduction takes place. This deduction scheme is
0000   ;           consistant but incorrect, and I have changed it by modifying
0000   ;           DELETE_LNMB and LNM$INSLOGTAB as follows:
0000   ;
0000   ;           When a logical name table is created, any quota explictely
0000   ;           allocated to the new table is deducted from the parent table's
0000   ;           quota holder as was previously being done; however, the size
0000   ;           of the new table itself is deducted from the quota holder of the
0000   ;           table that contains it (either the system or process directory
0000   ;           table). This is consistent with how logical names are handled,
0000   ;           and the philosophy that logical name tables are just logical
0000   ;           names with a special translation. When a logical name table is
0000   ;           deleted, the reverse of these two deductions takes place.
0000   ;
```

```
0000   ;            I have also removed the code for LNM$INSLOGN, LNM$TRANSLOGNAME,
0000   ;            and LNM$TRANSLATE. These routines are never called, and have
0000   ;            been commented out for months.
0000   ;
0000   ;    V03-009 ACG0354          Andrew C. Goldstein,    12-Sep-1983  21:55
0000   ;            Change RWACCESS field in CHIP$ block to FLAGS field
0000   ;
0000   ;    V03-008 TMK0004          Todd M. Katz                     31-Aug-1983
0000   ;            At the present time logical name blocks in each hash bucket are
0000   ;            ordered first by logical name size, then by access mode, then
0000   ;            alphabetically, by logical name, and finally by containing
0000   ;            table. The ordering of logical name blocks by access mode
0000   ;            before alphabetically meant that a supervisor mode "d" logical
0000   ;            name block would be found in the same hash bucket before an
0000   ;            executive mode "c" logical name block, and the explicit search
0000   ;            for "c" with the starting access mode specified as user or
0000   ;            supervisor would stop when the "d" logical name block was
0000   ;            encountered before the "c" was seen. Thus, "c" would never be
0000   ;            located unless an explicit search for an executive mode "c"
0000   ;            was initiated. This represented an error and is basically due
0000   ;            to the fact that searches for an explicit logical name are
0000   ;            performed for names at the given and inner access modes, while
0000   ;            the search process itself will stop as soon as a logical name
0000   ;            is encountered that collates higher than the logical name being
0000   ;            searched for, regardless if the name being searched for exists
0000   ;            at an inner access mode then the logical name block that
0000   ;            terminates the search. To fix this problem I have changed how
0000   ;            logical name blocks are ordered within a hash bucket. Logical
0000   ;            name blocks are now ordered first by the size of the logical
0000   ;            name, then alphabetically by logical name, then by access mode
0000   ;            of the logical name, and finally by the containing table's
0000   ;            table header address.
0000   ;
0000   ;    V03-007 TMK0003          Todd M. Katz                     09-Aug-1983
0000   ;            If LNM$INIT_PROT has been called to initialize the CHIP
0000   ;            protection template for a group logical name table (as
0000   ;            signalled by the setting of the bit LNMTH$V_GROUP), then zero
0000   ;            the member portion of the owner UIC field of the CHIP template
0000   ;            so that the group table does not have an owner.
0000   ;
0000   ;            Modify the routine LNM$CHECK_PROT so that it will grant access
0000   ;            to a logical name table, even when the caller would otherwise be
0000   ;            refused access based upon SOGW access protection, in two special
0000   ;            circumstances. First, if the logical name table is a group
0000   ;            logical name table (LNMTH$V_GROUP is set), the caller has the
0000   ;            GRPNAM privilege, the group logical name table is the caller's
0000   ;            group logical name table, and the caller has requested R or W
0000   ;            access to the table only, then return success granting access
0000   ;            to the table. Second if the logical name table is the system
0000   ;            logical name table (LNMTH$V_SYSTEM is set), the caller has the
0000   ;            SYSNAM privilege, and the caller has requested R or W access to
0000   ;            the table only, then return success granting access to the
0000   ;            table. These changes are required for compatibility reasons so
0000   ;            that access to the system and group tables is governed by the
0000   ;            same privileges across releases.
0000   ;
0000   ;            Comment out the unused routine LNM$INSLOGN.
```

```
0000    ;
0000    ;       V03-006 RAS0165         Ron Schaefer            5-Jul-1983
0000    ;               Correct RAS0158 and RAS0160 to ignore the caller's
0000    ;               access mode in the CHIP protection checking since the
0000    ;               logical name code uses a non-standard interpretation of
0000    ;               access mode.
0000    ;
0000    ;       V03-005 LMP0125         L. Mark Pilant,         26-Jun-1983  21:54
0000    ;               Change all referneces of CHIP$B_ACCESSOR_MODE to be
0000    ;               CHIP$L_ACCESSOR_MODE.
0000    ;
0000    ;       V03-004 RAS0160         Ron Schaefer                    16-Jun-1983
0000    ;               Add access mode of table to CHIP block and move
0000    ;               performance measurement cell to LNM$SEARCHLOG.
0000    ;
0000    ;       V03-003 RAS0158         Ron Schaefer                    25-May-1983
0000    ;               Add protection checking support subroutines LNM$INIT_PROT
0000    ;               to initialize the CHIP block for a table and LNM$CHECK_PROT
0000    ;               to check references to a table.
0000    ;
0000    ;       V03-002 TMK0002         Todd M. Katz                    25-Apr-1983
0000    ;               Fix several more bugs in these subroutines. The most
0000    ;               significant fix is one to LNM$SEARCHLOG. As this routine
0000    ;               locates the logical name tables in the list of such tables,
0000    ;               it searches for the presence of the given logical name within
0000    ;               them until a match is found. If the current logical name table
0000    ;               in the list of tables is shareable, only the shareable name
0000    ;               space is looked at for a match; likewise, if the current logical
0000    ;               name table is process-private, only the process-private name
0000    ;               space is referenced. The code that was making the distinction
0000    ;               between these two cases was incorrect, the result being that
0000    ;               the process-private name space was always being searched.
0000    ;               Furthermore, the code was incorrect in that processing of the
0000    ;               current table should be skipped if there is no name in the
0000    ;               same name space as the table itself. This was not being done
0000    ;               and allowed the possibility for access violations to occur.
0000    ;
0000    ;               In addition, fix the quota check in LNM$INSLOGN. The branch
0000    ;               following this check was signed, and it should have been
0000    ;               unsigned.
0000    ;
0000    ;               Also, increase the maximum recursion depth from 8 to 10.
0000    ;
0000    ;       V03-001 TMK0001         Todd M. Katz                    25-Mar-1983
0000    ;               Fix several bugs in these subroutines:
0000    ;
0000    ;               1. The PCB is only required on calls to LNM$TRNSLOGNAME,
0000    ;                  LNM$TRANSLATE, LNM$LOCKR, LNM$LOCKW, and LNM$UNLOCK.
0000    ;                  Therefore, remove the requirement that it be present in R4
0000    ;                  from all routines except for the above mentioned five.
0000    ;
0000    ;               2. When allocating and filling in the Recursive Table Name
0000    ;                  Control Block within LNM$FIRSTTAB and within LNM$SEARCHLOG,
0000    ;                  use a MOVZWL instead of a MOVZBL to fill in the access mode
0000    ;                  and set the case control bit because the latter occupies
0000    ;                  the first bit of the second byte of the source operand.
0000    ;
```

```
0000  ;          3. The table header of the process or system directory table is
0000  ;             supposed to go in the TABID Field of the Name Translation
0000  ;             Block when a table name is to be looked up by calling
0000  ;             LNM$LOOKUP. This routine was placing t'e address of the table
0000  ;             instead of the address of the table header in this field.
0000  ;
0000  ;          4. I have restructured the routine LNM$TABLE as follows: This
0000  ;             routine now saves the address of the translation block it is
0000  ;             working on in the Recursive Table Control Block before it
0000  ;             looks up the translation string in the hash tables. The
0000  ;             one exception is the original table name which is looked
0000  ;             up without being saved. This allows continuation down the
0000  ;             translation blocks of eight logical name blocks until a
0000  ;             table is found as long as the names being looked up are in
0000  ;             fact found. Previously, this information saving was being
0000  ;             done incorrectly, and at the wrong time. This meant that
0000  ;             the routine could not pick up and continue with the next
0000  ;             translation block after the current translation thread
0000  ;             dead ended.
0000  ;
0000  ;          5. I have restructured the routine LNM$CONTSEARCH as follows.
0000  ;             As this routine is processing the LNMB Blocks within the
0000  ;             current hash bucket it keeps the address of the previous
0000  ;             LNMB within the THREAD cell of the Name Translation Control
0000  ;             Block while the address of the LNMB it is working on is
0000  ;             always in R1. Then whenever this routine exits, regardless
0000  ;             of the status it exits with, R1 will always contain the
0000  ;             address of the LNMB Block that stopped the search (or 0 if
0000  ;             the list was exhausted) and NT_L_THREAD contains the address
0000  ;             of the previous LNMB block. This provided sufficient
0000  ;             information to allow new LNMB blocks to be inserted in the
0000  ;             correct place regardless of where that place is.
0000  ;
0000  ;             In addition this routine was performing a CMPC3, and then
0000  ;             checking the state of a bit within a field pointed at by R3
0000  ;             to determine whether the comparison is to be made case
0000  ;             sensitive or blind. Of course, the CMPC3 changes R3, so this
0000  ;             check must be made using the original contents of R3 which
0000  ;             have been stored on the stack.
0000  ;
0000  ;          6. I have re-written LNM$INSLOGTAB. There were many problems
0000  ;             with the old routine including the inability to correctly
0000  ;             find the table header of the new table entry, an inability
0000  ;             to just map a new entry to an existing entry when CREATE_IF
0000  ;             had been specified, and the way in which new entries were
0000  ;             linked into the exisiting list of entries within a hash
0000  ;             hash bucket was completely incorrect.
0000  ;
0000  ;          7. I have have re-written the routine DELLNMB as DELETE_LNMB,
0000  ;             the routine DELTAB as DELETE_NAMES, the routine LNM$DELETE,
0000  ;             and the routine LNM$DELTAB as LNM$DELETE_TAB. I have changed
0000  ;             the name of DEL1 to DELETE_ENTRY, added the routine
0000  ;             DELETE_TABLE, and eliminated the routine LNM$DELHASH.
0000  ;
0000  ;          9. The status SS$_EXLNMQUOTA is returned instead of SS$_EXQUOTA.
0000  ;
0000  ;         10. Put the names of the routines on the subtitles.
```

```
0000     628 ;
0000     629 ;               11. The routine LNM$SEARCHLOG was incorrectly searching for a
0000     630 ;                   logical name. This routine first ascertains that the logical
0000     631 ;                   name exists at all, and then searches among the possible
0000     632 ;                   logical name tables one-by-one until it finds the first
0000     633 ;                   table that contains such a logical name. The first time this
0000     634 ;                   routine encountered a process-private table, it would
0000     635 ;                   return success even if the table did not contain the logical
0000     636 ;                   name. This was due to a slight mistake in the ordering of
0000     637 ;                   some of the instructions which has been corrected.
0000     638 ;
0000     639 ;               12. Re-write LNM$INSLOGN.
0000     640 ;--
```

```
0000
0000          .SBTTL  DATA DEFINITIONS
0000
0000    ;
0000    ; MACRO LIBRARY CALLS:
0000    ;
0000
0000          $ARMDEF                               ;DEFINE ACCESS RIGHTS MASK
0000          $CADEF                                ;DEFINE CONDITIONAL ASSEMBLY SWITCHES
0000          $CHPCTLDEF                            ;DEFINE CHIP PROTECTION CONTROL BLOCK OFFSET
0000          $DYNDEF                               ;DEFINE STRUCTURE TYPE AND SUBTYPE CODES
0000          $LNMDEF                               ;DEFINE LOGICAL NAME ATTRIBUTES
0000          $LNMSTRDEF                            ;DEFINE LOGICAL NAME BLOCKS OFFSETS
0000          $ORBDEF                               ;DEFINE ORBJECT RIGHTS BLOCK OFFSETS
0000          $PCBDEF                               ;DEFINE PCB OFFSETS
0000          $PRVDEF                               ;DEFINE PRIVILEGE MASK OFFSETS
0000          $PSLDEF                               ;DEFINE PROCESSOR STATUS FIELDS
0000          $SSDEF                                ;DEFINE SYSTEM STATUS VALUES
0000
0000    ;
0000    ; ASSUMPTIONS ABOUT THE STRUCTURE AN OBJECT RIGHTS BLOCK AND A CHPCTL:
0000    ;
0000
0000          ASSUME  ORB$L_OWNER,          EQ,  0
0000          ASSUME  ORB$L_OWNER+4,        EQ,  ORB$L_ACL_MUTEX
0000          ASSUME  ORB$L_ACL_MUTEX+4,    EQ,  ORB$W_SIZE
0000          ASSUME  ORB$W_SIZE+2,         EQ,  ORB$B_TYPE
0000          ASSUME  ORB$B_TYPE+1,         EQ,  ORB$B_FLAGS
0000          ASSUME  ORB$B_FLAGS+3,        EQ,  ORB$W_REFCOUNT
0000          ASSUME  ORB$W_REFCOUNT+2,     EQ,  ORB$Q_MODE_PROT
0000          ASSUME  ORB$Q_MODE_PROT+8,    EQ,  ORB$L_SYS_PROT
0000          ASSUME  ORB$L_SYS_PROT+4,     EQ,  ORB$L_OWN_PROT
0000          ASSUME  ORB$L_OWN_PROT+4,     EQ,  ORB$L_GRP_PROT
0000          ASSUME  ORB$L_GRP_PROT+4,     EQ,  ORB$L_WOR_PROT
0000          ASSUME  ORB$L_WOR_PROT+4,     EQ,  ORB$L_ACL_COUNT
0000          ASSUME  ORB$L_ACL_COUNT+4,    EQ,  ORB$L_ACL_DESC
0000          ASSUME  ORB$L_ACL_DESC+4,     EQ,  ORB$R_MIN_CLASS
0000          ASSUME  ORB$R_MIN_CLASS+ORB$S_MIN_CLASS,-
0000                                        EQ,  ORB$R_MAX_CLASS
0000          ASSUME  ORB$R_MAX_CLASS+ORB$S_MAX_CLASS,-
0000                                        EQ,  ORB$K_LENGTH
0000
0000          ASSUME  CHPCTL$L_ACCESS,        EQ, 0
0000          ASSUME  CHPCTL$L_ACCESS+4,      EQ, CHPCTL$L_FLAGS
0000          ASSUME  CHPCTL$L_FLAGS+4,       EQ, CHPCTL$B_MODE
0000          ASSUME  CHPCTL$B_MODE+4,        EQ, CHPCTL$C_LENGTH
0000    ;     .PAGE
```

```
                 0000  ;*************************************************************************
                 0000  ;
                 0000  ; THIS CONTROL BLOCK IS ALSO LOCALLY DEFINED IN THE SHOW LOGICAL UTILITY
                 0000  ; ANY CHANGES TO THIS DATA STRUCTURE MUST BE REFLECTED IN SHOW LOGICAL AS WELL.
                 0000  ;
                 0000  ;*************************************************************************
                 0000
                 0000  ;
                 0000  ; BLOCK TO CONTROL RECURSIVE TABLE NAME TRANSLATION
                 0000  ;
                 0000  ;           31            25 24          16              9 8 7              0
                 0000  ;           +---------------+---------------+---------------+---------------+
                 0000  ; R5 ->     | TRIES         | DEPTH         |            T C| ACCESS MODE   |
                 0000  ;           +---------------+---------------+---------------+---------------+
                 0000  ;           |                ADDRESS OF NAME BLOCK                          |
                 0000  ;           +---------------+---------------+---------------+---------------+
                 0000  ;           |                                                               |
                 0000  ;           |        LNM$C_MAXDEPTH LONGWORDS TO SERVE AS STACK             |
                 0000  ;           |                                                               |
                 0000  ;           |                                                               |
                 0000  ;           +---------------+---------------+---------------+---------------+
                 0000  ;
                 0000  ;           F IS SET TO INHIBIT FURTHER RECURSION
                 0000  ;
       00000000  0000  RT_W_R5=        0
       00000000  0000  RT_B_ACMODE=    0
       00000001  0000  RT_B_FLAGS=     1
       00000008  0000  RT_V_CASE=      8                                 ;CASELESS FLAG
       00000001  0000  RT_M_CASE=      ^X1                               ;CASELESS MASK
       00000009  0000  RT_V_TERM=      9                                 ;INHIBIT RECURSION FLAG
       00000002  0000  RT_M_TERM=      ^X2                               ;INHIBIT RECURSION MASK
                 0000  ; BIT NUMBER 10 IS RESERVED.
                 0000
       00000002  0000  RT_B_DEPTH=     2                                 ;RECURSION DEPTH
       00000003  0000  RT_B_TRIES=     3                                 ;RECURSION TRIES
       000000FF  0000  RT_C_MAXTRIES=  255                               ;MAXIMUM NUMBER OF TRIES
       00000004  0000  RT_L_CACHEPTR=  4                                 ;POINTER TO CACHE ENTRY
       00000008  0000  RT_L_STACK=     8
       00000030  0000  RT_K_LENGTH=    RT_L_STACK+<4*LNM$C_MAXDEPTH>   ;BLOCK LENGTH
                 0000
                 0000  ;         .PAGE
```

```
                    0000  ;****************************************************************************
                    0000  ;
                    0000  ; THIS CONTROL BLOCK IS ALSO LOCALLY DEFINED IN THE SHOW LOGICAL UTILITY
                    0000  ; ANY CHANGES TO THIS DATA STRUCTURE MUST BE REFLECTED IN SHOW LOGICAL AS WELL.
                    0000  ;
                    0000  ;****************************************************************************
                    0000
                    0000  ;
                    0000  ; BLOCK TO CONTROL NAME TRANSLATION
                    0000  ;
                    0000  ;         31          25 24         16 15        A  8 7            0
                    0000  ;        +-------------+-------------+-------------+-------------+
                    0000  ; R3 ->  | HASH FUNCTION VALUE OR 0  |           M  C| ACCESS MODE |
                    0000  ;        +-------------+-------------+-------------+-------------+
                    0000  ;        |                     LENGTH OF NAME                    |
                    0000  ;        +-------------+-------------+-------------+-------------+
                    0000  ;        |                    ADDRESS OF NAME                    |
                    0000  ;        +-------------+-------------+-------------+-------------+
                    0000  ;        |               TABLE HEADER ADDRESS OR 0               |
                    0000  ;        +-------------+-------------+-------------+-------------+
                    0000  ;        |                NAME BLOCK ADDRESS OR 0                |
                    0000  ;        +-------------+-------------+-------------+-------------+
                    0000  ;
                    0000  ;
                    0000
          00000000  0000  NT_W_R5=        0
          00000000  0000  NT_B_ACMODE=    0
          00000001  0000  NT_B_FLAGS=     1
          00000001  0000  NT_M_CASE=      1                         ;CASELESS MASK
          00000008  0000  NT_V_CASE=      8                         ;CASELESS FLAG
                    0000  ; BIT NUMBER 9 IS RESERVED.
          00000004  0000  NT_M_MODIFY=    4                         ;HASH BUCKET SUSCEPTIBLE TO CHANGE MASK
          0000000A  0000  NT_V_MODIFY=    10                        ;HASH BUCKET SUSCEPTIBLE TO CHANGE FLAG
                    0000
          00000002  0000  NT_W_HASH=      2                         ;HASH FUNCTION OR 0
          00000004  0000  NT_L_NAMLEN=    4                         ;LENGTH OF NAME
          00000008  0000  NT_L_NAMADR=    8                         ;ADDRESS OF NAME
          0000000C  0000  NT_L_TABID=     12                        ;TABLE HEADER ADDRESS OR 0
          00000010  0000  NT_L_THREAD=    16                        ;LNMB POINTER
          00000014  0000  NT_K_LENGTH=    20                        ;BLOCK LENGTH
                    0000
                    0000  ;       .PAGE
```

```
                        0000             .SBTTL  DELETE_ENTRY    - DELETE ONE LOGICAL NAME TABLE ENTRY
                        0000     ;+
                        0000     ; DELETE_ENTRY - DELETE ONE LOGICAL NAME TABLE ENTRY
                        0000     ;
                        0000     ; THIS ROUTINE IS CALLED TO DELETE ONE LOGICAL NAME TABLE ENTRY.
                        0000     ; IF THE ENTRY IS A TABLE HEADER, IT IS LINKED TO R5.
                        0000     ; OTHERWISE, THE LOGICAL NAME BLOCK IS DELETED.
                        0000     ; IN EITHER CASE, ALL SPECIAL INFORMATION (OTHER THAN THE TABLE
                        0000     ; HEADER) IS HANDLED.
                        0000     ;
                        0000     ; INPUTS:
                        0000     ;
                        0000     ;       R1 = ADDRESS OF ENTRY TO BE DELETED.
                        0000     ;       R5 = ADDRESS OF LIST OF LOGICAL NAME BLOCKS CONTAINING
                        0000     ;            TABLE HEADERS
                        0000     ;
                        0000     ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR WRITE ACCESS,
                        0000     ;       AND THAT THE CALLER HAS THE PRIVILEGE OF DELETING THE LOGICAL NAME
                        0000     ;       TABLE ENTRY.
                        0000     ;
                        0000     ; OUTPUTS:
                        0000     ;
                        0000     ;       THE ENTRY IS REMOVED FROM ITS RESPECTIVE LOGICAL NAME TABLE AND THE
                        0000     ;       STORAGE IS RETURNED TO THE APPROPRIATE ALLOCATION REGION.
                        0000     ;
                        0000     ;       ADDITIONAL LOGICAL NAME BLOCKS CONTAINING TABLE HEADERS MAY BE
                        0000     ;       LINKED TO R5.
                        0000     ;       R0, R1, R2, AND R3 ARE MODIFIED.
                        0000     ;-
                        0000
                    00000000             .PSECT  YF$$LNM
                        0000     DELETE_ENTRY:                                  ;DELETE LOGICAL NAME TABLE ENTRY
         52     61  7D  0000             MOVQ    LNMB$L_FLINK(R1),R2            ;PICK BOTH LINKS
         63     82  DE  0003             MOVAL   (R2)+,@LNMB$L_FLINK(R3)        ;STORE NEXT PTR IN PREVIOUS BLOCK
                03  13  0006             BEQL    5$                             ;THIS IS THE END OF THE LINE
         62     53  D0  0008             MOVL    R3,LNMB$L_BLINK-4(R2)          ;STORE PREVIOUS PTR IN NEXT BLOCK
   53     11    A1  9E  000B     5$:     MOVAB   LNMB$T_NAME(R1),R3             ;POINT TO COUNTED NAME STRING
         50     83  9A  000F     10$:    MOVZBL  (R3)+,R0                       ;LENGTH OF NAME
         53     50  C0  0012             ADDL2   R0,R3                          ;ADDRESS OF TRANSLATION
   05    63     02  E0  0015             BBS     #LNMX$V_XEND, -                ;END OF TRANSLATIONS
                        0019                     LNMX$B_FLAGS(R3),20$
17 01    A3     07  E0  0019             BBS     #7,LNMX$B_INDEX(R3),70$ ;BRANCH IF SPECIAL TRANSLATION
         51     55  D1  001E     20$:    CMPL    R5,R1                          ;BLOCK LINKED ON TABLE LIST?
                2C  12  0021             BNEQ    DELETE_LNMB                    ;BRANCH IF NOT TO LINKED TO
                    05  0023             RSB                                    ;DELETE THE NAME BLOCK AND RETURN
                        0024
   50    05     A3  DE  0024     50$:    MOVAL   LNMX$T_XLATION+1(R3),R0 ;TRANSLATION STRING ADDRESS
                60  D5  0028             TSTL    (R0)                           ;IS ADDRESS PRESENT?
                04  13  002A             BEQL    60$                            ;NO ADDRESS SPECIFIED
                90  D4  002C             CLRL    @(R0)+                         ;CLEAR POINTER TO NAME
                70  D4  002E             CLRL    -(R0)                          ;CLEAR BACK POINTER
         53     04  C0  0030     60$:    ADDL    #LNMX$T_XLATION,R3             ;ADDRESS OF COUNTED TRANSLATION STRING
                DA  11  0033             BRB     10$                            ;PROCESS NEXT TRANSLATION
                        0035
81 8F    01     A3  91  0035     70$:    CMPB    LNMX$B_INDEX(R3),#LNMX$C_BACKPTR ;LOOK FOR BACK POINTER
                E8  13  003A             BEQL    50$                            ;PROCESS REFERENCE POINTER
82 8F    01     A3  91  003C             CMPB    LNMX$B_INDEX(R3),#LNMX$C_TABLE  ;LOOK FOR TABLE HEADER
```

```
             ED   12  0041   832         BNEQ    60$                        ;CONTINUE SCANNING TRANSLATIONS
                      0043   833         ASSUME  LNMB$L_FLINK,EQ,0
          81  55  D0  0043   834         MOVL    R5,(R1)+                   ;LINK ONTO R5 LIST
     61    05 A3  9E  0046   835         MOVAB   LNMX$T_XLATION+1(R3), -    ;STORE TABLE IDENTIFIER IN A HANDY PLACE
                      004A   836                 LNMB$L_BLINK-4(R1)
          55  71  DE  004A   837         MOVAL   -(R1),R5
             E1   11  004D   838         BRB     60$                        ;CONTINUE SCAN OF TRANSLATIONS
                      004F   839
                      004F   840 ;       .PAGE
```

```
                         004F          .SBTTL  DELETE_LNMB     - DELETE LOGICAL NAME BLOCK
                         004F   ;+
                         004F   ; DELETE_LNMB - DELETE LOGICAL NAME BLOCK
                         004F   ;
                         004F   ; THIS ROUTINE IS CALLED TO DELETE A LOGICAL NAME BLOCK.  STORAGE IS
                         004F   ; RETURNED TO THE APPROPRIATE ALLOCATION REGION AND QUOTA IS RETURNED
                         004F   ; TO THE APPROPRIATE TABLE HEADER(S).  IF THE LOGICAL NAME BLOCK IS FOR A
                         004F   ; SHAREABLE LOGICAL NAME TABLE, THEN THE OBJECT RIGHTS BLOCK ASSOCIATED WITH
                         004F   ; THE TABLE IS CLEANED UP BEFORE IT, TOGETHER WITH THE LNMB, ARE DELETED.
                         004F   ;
                         004F   ; INPUTS:
                         004F   ;
                         004F   ;       R1 = ADDRESS OF ENTRY TO BE DELETED.
                         004F   ;
                         004F   ;       IF THE ENTRY IS FOR A LOGICAL NAME TABLE, IT IS ASSUMED THAT THE ADDRESS
                         004F   ;       OF ITS TABLE HEADER IS STORED IN LNMB$L_BLINK.
                         004F   ;
                         004F   ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR WRITE ACCESS,
                         004F   ;       AND THAT THE CALLER HAS THE PRIVILEGE OF DELETING THE LOGICAL NAME
                         004F   ;       BLOCK.
                         004F   ;
                         004F   ; OUTPUTS:
                         004F   ;
                         004F   ;       STORAGE IS RETURNED TO THE APPROPRIATE ALLOCATION REGION.
                         004F   ;       THE ORB ASSOCIATED WITH THE LNMB, IF THERE IS ONE, IS CLEANED UP.
                         004F   ;       QUOTA IS RETURNED TO THE APPROPRIATE TABLE HEADER(S).
                         004F   ;
                         004F   ;       R0, R1, R2, AND R3 ARE MODIFIED.
                         004F   ;-
                         004F
                         004F   DELETE_LNMB:                                    ;DELETE LOGICAL NAME BLOCK
      50    51     D0    004F           MOVL    R1,R0                           ;ADDRESS OF LOGICAL NAME BLOCK
            03     E1    0052           BBC     #LNMB$V_TABLE,-                 ;LOGICAL NAME TABLE?
   44 10 A0              0054                   LNMB$B_FLAGS(R0),40$            ;NO - GO RETURN QUOTA FOR THE ENTRY
                         0057
                         0057   ;
                         0057   ; THE LOGICAL NAME BLOCK BEING DEALLOCATED IS FOR A LOGICAL NAME TABLE.  TAKE
                         0057   ; THE FOLLOWING ACTIONS BEFORE DELETING THE LOGICAL NAME BLOCK:
                         0057   ;
                         0057   ; 1. REMOVE THE LOGICAL NAME TABLE ENTRY BEING DELETED FROM THE HIERARCHY OF
                         0057   ;    LOGICAL NAME TABLES BY MODIFYING THE CHILD POINTER OF THE LOGICAL NAME
                         0057   ;    TABLE'S PARENT TABLE, AND THE SIBLING POINTERS OF THE LOGICAL NAME TABLE'S
                         0057   ;    SIBLINGS AS REQUIRED.
                         0057   ;
                         0057   ; 2. RETURN TO THE PARENT LOGICAL NAME TABLE'S QUOTA HOLDER (WHICH MAY IN FACT
                         0057   ;    BE THE PARENT TABLE) ANY QUOTA EXPLICTELY ALLOCATED TO THE TABLE.
                         0057   ;
                         0057   ; 3. IF THE LOGICAL NAME TABLE IS SHAREABLE, CLEANUP THE OBJECT RIGHTS BLOCK
                         0057   ;    ASSOCIATED WITH THE TABLE. THE ORB ITSELF WAS ALLOCATED CONTIGIOUSLY WITH
                         0057   ;    THE LOGICAL NAME BLOCK AND WILL BE DELETED WHEN THE LOGICAL NAME BLOCK IS
                         0057   ;    ITSELF DELETED.
                         0057   ;
                         0057
   53    04 A0     D0    0057           MOVL    LNMB$L_BLINK(R0),R3             ;ADDRESS OF TABLE HEADER
   52    0D A3     D0    005B           MOVL    LNMTH$L_PARENT(R3),R2           ;ADDRESS OF PARENT'S TABLE HEADER
   11 A2    53     D1    005F           CMPL    R3,LNMTH$L_CHILD(R2)            ;IS BLOCK THE IMMEDIATE CHILD OF PARENT?
            07     12    0063           BNEQ    10$                             ;NO  - GO FIND PRECEEDING SIBLING
```

```
          15 A3   D0  0065              MOVL    LNMTH$L_SIBLING(R3),-     ;YES - BLOCK'S IMMEDIATE SIBLING BECOMES
          11 A2       0068                      LNMTH$L_CHILD(R2)         ;       PARENT'S IMMEDIATE CHILD
             15   11  006A              BRB     30$                       ;GO RETURN DEDUCTED QUOTA
                      006C
   51     11 A2   D0  006C      10$:    MOVL    LNMTH$L_CHILD(R2),R1      ;ADDRESS OF PARENT'S IMMEDIATE CHILD
   15 A1     53   D1  0070      15$:    CMPL    R3,LNMTH$L_SIBLING(R1)    ;DOES THE SIBLING PRECEED THE BLOCK?
             06   13  0074              BEQL    20$                       ;YES - GO UNHOOK IT FROM LIST
   51     15 A1   D0  0076              MOVL    LNMTH$L_SIBLING(R1),R1    ;NO  - RETRIEVE ADDRESS OF NEXT SIBLING
             F4   11  007A              BRB     15$                       ;       AND CONTINUE SEARCH
                      007C
          15 A3   D0  007C      20$:    MOVL    LNMTH$L_SIBLING(R3),-     ;UNHOOK LOGIAL NAME BLOCK FROM THE LIST
          15 A1       007F                      LNMTH$L_SIBLING(R1)       ;OF SIBLINGS
                      0081
   52     19 A2   D0  0081      30$:    MOVL    LNMTH$L_QTABLE(R2),R2     ;RETRIEVE PARENT'S QUOTA HOLDER
          1D A3   C0  0085              ADDL2   LNMTH$L_BYTESLM(R3),-     ;RETURN ANY DEDUCTED QUOTA TO PARENT'S
          21 A2       0088                      LNMTH$L_BYTES(R2)         ;QUOTA HOLDER
                      008A
   51     05 A3   D0  008A              MOVL    LNMTH$L_ORB(R3),R1        ;RETRIEVE TABLE'S OBJECT RIGHTS BLOCK
             0B   13  008E              BEQL    40$                       ;SKIP ORB CLEANUP IF THERE ISN'T ONE
             50   DD  0090              PUSHL   R0                        ;SAVE LNMB ADDRESS
   00000000'EF    16  0092              JSB     EXE$CLEANUP_ORB           ;CLEANUP THE ORB
          50 8ED0     0098              POPL    R0                        ;RESTORE LNMB ADDRESS
                      009B
                      009B      ;
                      009B      ; FINALLY, RETURN THE SIZE OF THE LOGICAL NAME (OR LOGICAL NAME TABLE) TO THE
                      009B      ; QUOTA HOLDER OF THE CONTAINING TABLE, AND THEN PERFORM THE DELETION OF THE
                      009B      ; ACTUAL BLOCK.
                      009B      ;
                      009B
   51     08 A0   3C  009B      40$:    MOVZWL  LNMB$W_SIZE(R0),R1        ;SIZE OF LOGICAL NAME BLOCK TO RETURN
   52     0C A0   D0  009F              MOVL    LNMB$L_TABLE(R0),R2       ;ADDRESS OF CONTAINING TABLE'S HEADER
                      00A3
                      00A3      ;
                      00A3      ; BUMP THE APPROPRIATE DIRECTORY SEQUENCE NUMBER IF THE CONTAINING TABLE
                      00A3      ; IS ONE OF THE DIRECTORY TABLES.
                      00A3      ;
                      00A3
             01   E1  00A3              BBC     #LNMTH$V_DIRECTORY,-      ;DIRECTORY TABLE?
          12 62       00A5                      LNMTH$B_FLAGS(R2),60$     ;OKAY IF NOT
   08 52     1F   E0  00A7              BBS     #31,R2,50$                ;BRANCH IF SYSTEM DIRECTORY
   00000000'9F    D6  00AB              INCL    @#CTL$GL_LNMDIRSEQ        ;BUMP PROCESS DIRECTORY COUNTER
             06   11  00B1              BRB     60$
   00000000'9F    D6  00B3      50$:    INCL    @#LNM$GL_SYSDIRSEQ        ;BUMP SYSTEM DIRECTORY COUNTER
                      00B9
   52     19 A2   D0  00B9      60$:    MOVL    LNMTH$L_QTABLE(R2),R2     ;ADDRESS OF QUOTA HOLDER OF TABLE
   21 A2     51   C0  00BD              ADDL2   R1,LNMTH$L_BYTES(R2)      ;RETURN SIZE OF LOGICAL NAME
           00CE   31  00C1              BRW     LNM$DELBLK                ;DELETE LOGICAL NAME BLOCK AND RETURN
                      00C4
                      00C4      ;       .PAGE
                      00C4       .SBTTL  DELETE_NAMES    - SCAN HASH TABLE AND DELETE NAMES
                      00C4  ;+
                      00C4  ; DELETE_NAMES - SCAN HASH TABLE AND DELETE NAMES
                      00C4  ;
                      00C4  ; THIS ROUTINE SCANS A HASH TABLE AND DELETES ALL OF THE NAMES IN A SPECIFIED
                      00C4  ; TABLE WITH AN ACCESS MODE GREATER THAN OR EQUAL TO THE SPECIFIED ACCESS MODE.
                      00C4  ;
                      00C4  ; THIS ROUTINE IS SET UP SO THAT IT WILL NEVER DELETE A DIRECTORY.
                      00C4  ;
                      00C4  ; INPUTS:
                      00C4  ;
                      00C4  ;       R1 = ADDRESS OF TABLE HEADER OR 0 FOR ALL TABLES.
                      00C4  ;       R2 = ACCESS MODE.
                      00C4  ;       R3 = ADDRESS OF HASH TABLE TO BE SCANNED.
                      00C4  ;
                      00C4  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR WRITE ACCESS,
                      00C4  ;       AND THAT THE CALLER HAS THE PRIVILEGE OF DELETING THE LOGICAL NAME
                      00C4  ;       TABLE ENTRIES WITHIN THE SPECIFIED TABLE.
                      00C4  ;
                      00C4  ; OUTPUTS:
                      00C4  ;
                      00C4  ;       ENTRIES ARE REMOVED FROM THEIR RESPECTIVE LOGICAL NAME TABLES AND THE
                      00C4  ;       STORAGE IS RETURNED TO THE APPROPRIATE ALLOCATION REGION.
                      00C4  ;
                      00C4  ;       R1, R2, R3, R4, and R5 ARE MODIFIED.
                      00C4  ;-
                      00C4
                      00C4  DELETE_NAMES:
               55  D4 00C4          CLRL    R5                      ;ZERO LINKED LIST OF TABLE HEADERS
           54  51  D0 00C6          MOVL    R1,R4                   ;ADDRESS OF TABLE HEADER
           50  63  CE 00C9          MNEGL   LNMHSH$L_MASK(R3),R0    ;MAXIMUM BUCKET NUMBER
     53  0C  A3  DE 00CC          MOVAL   LNMHSH$K_BUCKET(R3),R3  ;BUCKET ADDRESS
           51  83  D0 00D0  10$:    MOVL    (R3)+,R1                ;BEGINNING OF HASH CHAIN
               28  13 00D3          BEQL    40$                     ;EMPTY CHAIN
               54  D5 00D5  20$:    TSTL    R4                      ;SPECIFIC TABLE
               06  13 00D7          BEQL    25$                     ;NOPE
        0C A1  54  D1 00D9          CMPL    R4,LNMB$L_TABLE(R1)     ;RIGHT TABLE?
               19  12 00DD          BNEQ    30$                     ;BRANCH IF TABLES DON'T MATCH
        0B A1  52  91 00DF  25$:    CMPB    R2,LNMB$B_ACMODE(R1)    ;ACCESS MODE COMPATIBLE?
               13  1A 00E3          BGTRU   30$                     ;BRANCH IF NOT
               04  E0 00E5          BBS     #LNMB$V_NODELETE,-      ;DIRECTORY (NODELETE WILL BE SET)?
         0E 10 A1    00E7                  LNMB$B_FLAGS(R1),30$    ;BRANCH IF DIRECTORY, NEVER DELETE ONE
               1F  BB 00EA          PUSHR   #^M<R0,R1,R2,R3,R4>     ;SAVE SOME REGISTERS
   04 AE   04  A1  D0 00EC          MOVL    LNMB$L_BLINK(R1),4(SP)  ;SAVE ADDRESS OF PREVIOUS BLOCK
             FF0C  30 00F1          BSBW    DELETE_ENTRY            ;DELETE LOGICAL NAME BLOCK
               0B  10 00F4          BSBB    DELETE_TABLE            ;DELETE TABLE IF CURRENT LNMB IS ONE
               1F  BA 00F6          POPR    #^M<R0,R1,R2,R3,R4>     ;RESTORE REGISTERS
           51  61  D0 00F8  30$:    MOVL    LNMB$L_FLINK(R1),R1     ;MOVE TO NEXT LNMB IN HASH BUCKET
               D8  12 00FB          BNEQ    20$                     ;BRANCH IF IT EXISTS
        D0 50  F5     00FD  40$:    SOBGTR  R0,10$                  ;BUMP TO NEXT HASH BUCKET
                   05 0100          RSB
                      0101
                      0101  ;       .PAGE
                      0101        .SBTTL  DELETE_TABLE     - DELETE A LOGICAL NAME TABLE
                      0101  ;+
                      0101  ; DELETE_TABLE - DELETE A LOGICAL NAME TABLE
                      0101  ;
                      0101  ; THIS ROUTINE TAKES A TABLE HEADER AND DELETES IT. TO DELETE A TABLE HEADER,
                      0101  ; FIRST ALL OF THE CHILDREN OF THE TABLE ARE DELETED BY RECURSIVELY CALLING
                      0101  ; THIS ROUTINE, THEN ALL OF THEN NAMES DEFINED WITHIN THE TABLE ARE DELETED,
                      0101  ; AND FINALLY THE LOGICAL NAME BLOCK CONTAINING THE TABLE HEADER ITSELF IS
                      0101  ; DELETED.
                      0101  ;
                      0101  ; INPUTS:
                      0101  ;
                      0101  ;       R5 = ADDRESS OF LIST OF LOGICAL NAME BLOCK CONTAINING TABLE HEADERS TO
                      0101  ;            BE DELETED.
                      0101  ;
                      0101  ;       IT IS ASSUMED THAT THE ADDRESS OF THE TABLE HEADER IS STORED WITHIN
                      0101  ;       LNMB$L_BLINK(R5).
                      0101  ;
                      0101  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR WRITE ACCESS,
                      0101  ;       AND THAT THE CALLER HAS THE PRIVILEGE OF DELETING THE LOGICAL NAME
                      0101  ;       TABLE.
                      0101  ;
                      0101  ; OUTPUTS:
                      0101  ;
                      0101  ;       ALL OF THE NAMES DEFINED WITHIN THE TABLE HEADER ARE DELETED.
                      0101  ;       ALL OF THE CHILDREN OF THE TABLE HEADER ARE DELETED (THIS INVOLVES
                      0101  ;       DELETING ALL OF THEIR CHILDREN, AND ALL OF THE NAMES DEFINED WITHIN
                      0101  ;       THEM ETC...).
                      0101  ;       STORAGE AND QUOTA IS RETURNED.
                      0101  ;
                      0101  ;       R1, R2, R3, R4, and R5 ARE MODIFIED.
                      0101  ;-
                      0101
                      0101 DELETE_TABLE:
      50    55   D0   0101        MOVL    R5,R0                   ;ADDRESS OF CURRENT TABLE HEADER'S LNMB
            04   12   0104        BNEQ    10$                     ;BRANCH IF ONE PRESENT
      50    01   D0   0106        MOVL    #SS$_NORMAL,R0          ;RETURN SUCCESS WHEN ALL DONE
                 05   0109        RSB
                      010A
                      010A ;
                      010A ; DELETE ALL OF THE CHILDREN OF THE CURRENT TABLE HEADER.
                      010A ;
                      010A
   51    04 A0   D0   010A 10$:   MOVL    LNMB$L_BLINK(R0),R1     ;ADDRESS OF TABLE HEADER
   52    11 A1   D0   010E        MOVL    LNMTH$L_CHILD(R1),R2    ;ADDRESS OF CHILD TABLE
            09   13   0112        BEQL    20$                     ;BRANCH IF NO CHILD
   51    09 A2   D0   0114        MOVL    LNMTH$L_NAME(R2),R1     ;ADDRESS OF NAME BLOCK OF CHILD
          FEE5   30   0118        BSBW    DELETE_ENTRY            ;DELETE THE CHILD (PUT IT ON THE LIST)
            E4   11   011B        BRB     DELETE_TABLE            ;CONTINUE THE DELETION PROCESS
                      011D
                      011D ;
                      011D ; DELETE ALL OF THE NAMES DEFINED WITHIN THE CURRENT TABLE HEADER.
                      011D ;
                      011D
            60   DD   011D 20$:   PUSHL   LNMB$L_FLINK(R0)        ;UPDATE AND SAVE LIST POINTER
            50   DD   011F        PUSHL   R0                      ;SAVE CURRENT LOGICAL NAME BLOCK ADDRESS
   53    01 A1   D0   0121        MOVL    LNMTH$L_HASH(R1),R3     ;HASH TABLE ADDRESS
```

```
      52   D4  0125  1060          CLRL    R2                      ;ALL ACCESS MODES
    FF9A   30  0127  1061          BSBW    DELETE_NAMES            ;DELETE NAMES IN THIS TABLE
      22   BA  012A  1062          POPR    #^M<R1,R5>              ;RESTORE NAME BLOCK ADDRESS AND LIST PTR
    FF20   30  012C  1063          BSBW    DELETE_LNMB             ;DELETE THE CURRENT LOGICAL NAME BLOCK
      D0   11  012F  1064          BRB     DELETE_TABLE            ;ITERATE UNTIL LIST IS EMPTY
               0131  1065
               0131  1066 ;        .PAGE
```

```
                    0131          .SBTTL  LNM$CHECK_PROT  - CHECK ACCESS TO A LOGICAL NAME TABLE
                    0131   ;+
                    0131   ; LNM$CHECK_PROT - CHECK ACCESS TO A LOGICAL NAME TABLE
                    0131   ;
                    0131   ; THIS ROUTINE IS CALLED TO CHECK ACCESS TO A LOGICAL NAME TABLE, USING ITS
                    0131   ; OBJECT RIGHTS BLOCK.
                    0131   ;
                    0131   ; ASSUMPTION: THE LOGICAL NAME TABLE HAS AN ASSOCIATED OBJECT RIGHTS BLOCK
                    0131   ;               IE - LNMTH$L_CHP(R1)  = 0!
                    0131   ;
                    0131   ; CURRENTLY, ONLY SOGW PROTECTION IS ACTUALLY CHECKED.   ACCESS MODE
                    0131   ; CHECKING IS HANDLED BY THE CALLING ROUTINE AND ALL OTHER SECURITY
                    0131   ; CHECKS ARE NYI.
                    0131   ;
                    0131   ; INORDER TO PROVIDE COMPATIBLE ACCESS TO THE GROUP AND SYSTEM LOGICAL NAME
                    0131   ; TABLES ACROSS RELEASES, THE FOLLOWING ALGORITHM (WHICH WILL ALLOW ACCESS
                    0131   ; UNDER SPECIAL SETS OF CIRCUMSTANCES TO THESE TABLES EVEN IF ACCESS IS DENIED
                    0131   ; BY SOGW PROTECTION) IS IMPLEMENTED:
                    0131   ;
                    0131   ; IF SOGW PROTECTION
                    0131   ; THEN
                    0131   ;     RETURN SUCCESS
                    0131   ; ELSE
                    0131   ;     IF OTHER THAN R OR W ACCESS IS REQUESTED
                    0131   ;     THEN
                    0131   ;         RETURN FAILURE
                    0131   ;     ELSE
                    0131   ;         IF GROUP LOGICAL NAME TABLE
                    0131   ;         THEN
                    0131   ;             IF GRPNAM AND TABLE IS GROUP TABLE FOR THE CALLER
                    0131   ;             THEN
                    0131   ;                 RETURN SUCCESS
                    0131   ;             ELSE
                    0131   ;                 RETURN FAILURE
                    0131   ;         ELSE
                    0131   ;             IF SYSTEM LOGICAL NAME TABLE AND SYSNAM
                    0131   ;             THEN
                    0131   ;                 RETURN SUCCESS
                    0131   ;             ELSE
                    0131   ;                 RETURN FAILURE
                    0131   ;
                    0131   ; INPUTS:
                    0131   ;
                    0131   ;       R1 = ADDRESS OF LOGICAL NAME TABLE HEADER.
                    0131   ;       R2 = ACCESS MASK TO USE FOR THE CHECK.
                    0131   ;       R4 = CURRENT PCB ADDR
                    0131   ;
                    0131   ; OUTPUTS:
                    0131   ;
                    0131   ;       R0 = RETURN STATUS FROM THE EXE$CHKPRO SUBROUTINE.
                    0131   ;            (SUCCESS IN TWO SPECIAL CASES - SEE ABOVE)
                    0131   ;       R2 IS DESTROYED. ALL OTHER REGISTERS PRESERVED.
                    0131   ;-
                    0131
                    0131   LNM$CHECK_PROT::                          ;CHECK ACCESS TO A LOGICAL NAME TABLE
          53   DD   0131          PUSHL   R3                        ;SAVE REGISTERS
    7E    51   7D   0133          MOVQ    R1,-(SP)
```

```
50      008C C4    D0  0136                MOVL    PCB$L_ARB(R4),R0            ;RETRIEVE AGENT RIGHTS BLOCK
51        05 A1    D0  013B                MOVL    LNMTH$L_ORB(R1),R1          ;RETRIEVE OBJECT RIGHTS BLOCK
                       013F
             7E    D4  013F                CLRL    -(SP)                       ;CLEAR ACCESS MODE FIELD OF CHPCTL
             05    DD  0141                PUSHL   #<CHPCTL$M_READ!-           ;SET READ AND USEREADALL BITS IN
                       0143                        CHPCTL$M_USEREADALL>        ;CHPCTL FLAGS FIELD
52        FE 8F    93  0143                BITB    #^C<ARM$M_READ>,R2          ;OTHER THAN READ ACCESS CHECK REQUESTED?
             03    13  0147                BEQL    5$                          ;BRANCH IF NOT
      6E     06    8C  0149                XORB2   #<CHPCTL$M_WRITE!-          ;OTHERWISE SET WRITE AND CLEAR
                       014C                        CHPCTL$M_USEREADALL>,(SP)   ;USEREADLL BITS IN CHPCTL FLAGS FIELD
             52    DD  014C        5$:     PUSHL   R2                          ;SET ACCESS REQUESTED IN CHPCTL
                       014E
      52     5E    D0  014E                MOVL    SP,R2                       ;MOVE CHPCTL ADDRESS INTO PROPER REG
             53    D4  0151                CLRL    R3                          ;NO CHPRET ACCESS BLOCK REQUIRED
  00000000'GF      16  0153                JSB     G^EXE$CHKPRO_INT            ;CHECK THE PROTECTION
  5E      0C A2    9E  0159                MOVAB   CHPCTL$C_LENGTH(R2),SP      ;REMOVE CHPCTL BLOCK FROM THE STACK
      51     8E    7D  015D                MOVQ    (SP)+,R1                    ;RESTORE REGISTERS
             53  8ED0  0160                POPL    R3
      2B     50    E8  0163                BLBS    R0,30$                      ;RETURN IF OK TO ACCESS
                       0166
52        FC 8F    93  0166                BITB    #^C<ARM$M_READ!-            ;OTHER THAN R OR W ACCESS REQUESTED?
                       016A                        ARM$M_WRITE>,R2
             25    12  016A                BNEQ    30$                         ;IF SO THEN RETURN FAILURE
                       016C
             02    E1  016C                BBC     #LNMTH$V_GROUP,-            ;IS THIS A GROUP LOGICAL NAME TABLE?
          14 61        016E                        LNMTH$B_FLAGS(R1),10$       ;BRANCH IF IT ISN'T ONE
                       0170                IFNPRIV GRPNAM,30$                  ;RETURN FAILURE IF CALLER LACKS GRPNAM
52        05 A1    D0  0176                MOVL    LNMTH$L_ORB(R1),R2          ;RETRIEVE ADDRESS OF OBJECT RIGHTS BLOCK
        00BE C4    B1  017A                CMPW    PCB$L_UIC+2(R4),-           ;IS THIS THE CALLER'S GROUP LOGICAL
          02 A2        017E                        ORB$L_OWNER+2(R2)           ;NAME TABLE?
             0C    13  0180                BEQL    20$                         ;RETURN SUCCESS IF IT
             0D    11  0182                BRB     30$                         ;RETURN FAILURE IF IT ISN'T
                       0184
             03    E1  0184        10$:    BBC     #LNMTH$V_SYSTEM,-           ;IS THIS THE SYSTEM LOGICAL NAME TABLE
          09 61        0186                        LNMTH$B_FLAGS(R1),30$       ;RETURN FAILURE IF IT ISN'T
                       0188                IFNPRIV SYSNAM,30$                  ;RETURN FAILURE IF CALLER LACKS SYSNAM
                       018E
      50     01    D0  018E        20$:    MOVL    #SS$_NORMAL,R0              ;CHANGE STATUS OF ROUTINE TO SUCCESS
             05        0191        30$:    RSB                                 ;RETURN STATUS
                       0192
                       0192        ;       .PAGE
```

```
                      0192       .SBTTL  LNM$DELBLK      - DELETE P1 OR SO PAGED BLOCK
                      0192  ;+
                      0192  ; LNM$DELBLK - DELETE P1 OR SO PAGED BLOCK
                      0192  ;
                      0192  ; THIS ROUTINE IS CALLED TO DELETE A CHUNK OF P1 OR SO PAGED MEMORY.
                      0192  ;
                      0192  ; INPUTS:
                      0192  ;
                      0192  ;       R0 = ADDRESS OF ENTRY TO BE DELETED.
                      0192  ;
                      0192  ; OUTPUTS:
                      0192  ;
                      0192  ;       NONE
                      0192  ;
                      0192  ;       STORAGE IS RETURNED TO THE APPROPRIATE ALLOCATION REGION.
                      0192  ;
                      0192  ;       R0, R1, R2, AND R3 ARE MODIFIED.
                      0192  ;-
                      0192
                      0192  LNM$DELBLK::                            ;DELETE PAGED BLOCK
   51   08 A0    3C   0192          MOVZWL  LNMB$W_SIZE(R0),R1      ;SIZE OF BLOCK
   06 50    1F   E0   0196          BBS     #31,R0,10$              ;IF SET, SYSTEM SPACE TABLE
 00000000'9F     17   019A          JMP     @#EXE$DEAP1             ;DEALLOCATE PROCESS DYNAMIC MEMORY
 00000000'9F     17   01A0  10$:    JMP     @#EXE$DEAPAGED          ;DEALLOCATE SYSTEM PAGED MEMORY
                      01A6
                      01A6  ;       .PAGE
```

```
                         01A6            .SBTTL  LNM$DELETE      - DELETE LOGICAL NAME TABLE ENTRY
                         01A6   ;+
                         01A6   ; LNM$DELETE - DELETE LOGICAL NAME TABLE ENTRY
                         01A6   ;
                         01A6   ; THIS ROUTINE IS CALLED TO DELETE A LOGICAL NAME TABLE ENTRY, RETURN ITS
                         01A6   ; STORAGE TO THE APPROPRIATE ALLOCATION REGION, AND RETURN ITS QUOTA.
                         01A6   ; IF THE NAME IS A TABLE HEADER, THEN ALL NAMES IN THE TABLE ARE ALSO
                         01A6   ; DELETED AND ANY TABLES OF WHICH THIS TABLE IS THE PARENT ARE DELETED.
                         01A6   ;
                         01A6   ; A PRIVILEGE ERROR WILL ALWAYS BE RETURNED IF AN ATTEMPT IS MADE TO DELETE
                         01A6   ; A DIRECTORY.
                         01A6   ;
                         01A6   ; INPUTS:
                         01A6   ;
                         01A6   ;        R1 = ADDRESS OF ENTRY TO BE DELETED.
                         01A6   ;
                         01A6   ;        IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR WRITE ACCESS,
                         01A6   ;        AND THAT THE CALLER HAS THE PRIVILEGE OF DELETING THE LOGICAL NAME
                         01A6   ;        BLOCK.
                         01A6   ;
                         01A6   ; OUTPUTS:
                         01A6   ;
                         01A6   ;        R0 = SS$_NOPRIV IF THE LOGICAL NAME TABLE ENTRY IS A DIRECTORY.
                         01A6   ;
                         01A6   ;        R0 = SS$_SUCCESS
                         01A6   ;
                         01A6   ;        THE ENTRY IS REMOVED FROM ITS RESPECTIVE LOGICAL NAME TABLE AND THE
                         01A6   ;        STORAGE IS RETURNED TO THE APPROPRIATE ALLOCATION REGION. IF THE ENTRY
                         01A6   ;        HAS AN ASSOCIATED MAILBOX UCB ADDRESS, THEN THE LINKAGE FROM THE UCB
                         01A6   ;        TO THE LOGICAL NAME ENTRY IS CLEARED.
                         01A6   ;
                         01A6   ;        R1, R2, AND R3 ARE MODIFIED.
                         01A6   ;-
                         01A6
                         01A6
                         01A6   LNM$DELETE:                                     ;DELETE LOGICAL NAME TABLE ENTRY
             04     E1   01A6            BBC     #LNMB$V_NODELETE,-      ;DIRECTORY (NODELETE WILL BE SET)?
       05 10 A1          01A8                    LNMB$B_FLAGS(R1),10$    ;IF NOT THEN GO DELETE THE ENTRY
       50    24     3C   01AB            MOVZWL  #SS$_NOPRIV,R0          ;OTHERWISE RETURN A PRIVILEGE VIOLATION
             0E     11   01AE            BRB     20$
                         01B0
       7E    54     7D   01B0   10$:     MOVQ    R4,-(SP)                ;SAVE REGISTERS R4 AND R5
             55     D4   01B3            CLRL    R5                      ;CLEAR LINKED LIST OF TABLE HEADERS
           FE48     30   01B5            BSBW    DELETE_ENTRY            ;DELETE THE LOGICAL NAME TABLE ENTRY
           FF46     30   01B8            BSBW    DELETE_TABLE            ;DELETE THE TABLE HEADER IF IT IS ONE
       54    8E     7D   01BB            MOVQ    (SP)+,R4                ;RESTORE REGISTERS R4 AND R5
                    05   01BE   20$:     RSB                             ;RETURN STATUS
                         01BF
                         01BF   ;        .PAGE
```

```
                        01BF          .SBTTL  LNM$DELETE_LNMB - DELETE LOGICAL NAME TABLE ENTRY PLUS ALIASES
                        01BF  ;+
                        01BF  ; LNM$DELETE_LNMB - DELETE LOGICAL NAME TABLE ENTRY PLUS ALIASES
                        01BF  ;
                        01BF  ; THIS ROUTINE IS CALLED TO DELETE A LOGICAL NAME TABLE ENTRY TOGETHER WITH ALL
                        01BF  ; OF ITS OUTER ACCESS MODE ALIASES. STORAGE FOR THE DELETED ENTRIES IS RETURNED
                        01BF  ; TO THE APPROPRIATE ALLOCATION REGION, AND QUOTA IS RETURNED AS WELL. IF ANY
                        01BF  ; OF THE ENTRIES DELETED ARE LOGICAL NAME TABLES, THEN ALL NAMES WITHIN THE
                        01BF  ; TABLE ARE ALSO DELETED AS WELL AS ANY TABLES OF WHICH THIS TABLE IS THE
                        01BF  ; PARENT OF.
                        01BF  ;
                        01BF  ; INPUTS:
                        01BF  ;
                        01BF  ;       R1 = ADDRESS OF LOGICAL NAME TABLE ENTRY
                        01BF  ;
                        01BF  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR WRITE ACCESS,
                        01BF  ;       AND THAT THE CALLER HAS THE PRIVILEGE OF DELETING THE LOGICAL NAME
                        01BF  ;       TABLE ENTRY.
                        01BF  ;
                        01BF  ; OUTPUTS:
                        01BF  ;
                        01BF  ;       R0 = SS$_NOPRIV IF THE LOGICAL NAME TABLE ENTRY IS A DIRECTORY.
                        01BF  ;
                        01BF  ;            SS$_SUCCESS.
                        01BF  ;
                        01BF  ;            SS$_NOLOGNAM IF THERE ARE NO SUCH LOGICAL NAMES.
                        01BF  ;
                        01BF  ;       THE ENTRIES ARE REMOVED FROM THEIR LOGICAL NAME TABLE AND THE STORAGE
                        01BF  ;       IS RETURNED TO THE APPROPRIATE ALLOCATION REGION. IF ANY OF THE ENTRIES
                        01BF  ;       HAS AN ASSOCIATED MAILBOX UCB OR VOLUME UCB ADDRESS, THEN THE LINKAGE
                        01BF  ;       FROM THE UCB TO THE LOGICAL NAME TABLE ENTRY IS CLEARED BY THIS
                        01BF  ;       ROUTINE'S CALLER.
                        01BF  ;
                        01BF  ;       R1, R2, AND R3 ARE MODIFIED.
                        01BF  ;-
                        01BF
                        01BF LNM$DELETE_LNMB::                                  ;DELETE TABLE ENTRY AND ALL ALIASES
              7E   D4   01BF          CLRL    -(SP)                             ;INITIALIZE THREAD TO ZERO
           0C A1   DD   01C1          PUSHL   LNMB$L_TABLE(R1)                  ;ADDRESS OF TABLE HEADER
           12 A1   9F   01C4          PUSHAB  LNMB$T_NAME+1(R1)                 ;DESCRIPTOR OF LOGICAL NAME
      7E   11 A1   9A   01C7          MOVZBL  LNMB$T_NAME(R1),-(SP)
    00000403 8F    DD   01CB          PUSHL   #<NT_M_MODIFY @ 8+-               ;HASH BUCKET MIGHT BE MODIFIED
                        01D1                  PSL$C_USER>                       ;SEARCH ACCESS MODE
              51   DD   01D1          PUSHL   R1                                ;SAVE ADDRESS OF INNERMOST LNMB
                        01D3
              03   91   01D3          CMPB    #PSL$C_USER,-                     ;IS INNERMOST LOGICAL NAME TABLE ENTRY
           0B A1        01D5                  LNMB$B_ACMODE(R1)                 ;AN USER ACCESS MODE NAME?
              07   13   01D7          BEQL    10$                               ;IF SO, THEN IT IS ONLY ENTRY TO DELETE
      53   04 AE   DE   01D9          MOVAL   4(SP),R3                          ;ELSE SETUP TO FIND OUTERMOST MODE ENTRY
            02EB   30   01DD          BSBW    LNM$PRESEARCH                     ;AND GO FIND IT
                        01E0
              51   DD   01E0 10$:     PUSHL   R1                                ;SAVE ADDRESS OF CURRENT TABLE ENTRY
              C2   10   01E2          BSBB    LNM$DELETE                        ;DELETE CURRENT TABLE ENTRY
              51 8ED0   01E4          POPL    R1                                ;RESTORE ADDRESS OF DELETED ENTRY
        11    50   E9   01E7          BLBC    R0,30$                            ;RETURN ANY ERROR ON DELETION
                        01EA
      6E      51   D1   01EA          CMPL    R1,(SP)                           ;WAS INNERMOST ENTRY JUST DELETED?
```

```
           09   13  01ED            BEQL    20$                     ;IF SO THEN GO RETURN SUCCESS
  53    04 AE   DE  01EF            MOVAL   4(SP),R3                ;ELSE PREPARE TO SEARCH FOR NEXT ENTRY
         02FF   30  01F3            BSBW    LNM$CONTSEARCH          ;SEARCH FOR NEXT ENTRY TO DELETE
           E8   11  01F6            BRB     10$                     ;ELSE GO SEE IF ENTRY CAN BE DELETED
                    01F8
     50    01   D0  01F8   20$:     MOVL    #SS$_NORMAL,R0          ;SUCCESS
     5E    18   C0  01FB   30$:     ADDL2   #NT_K_LENGTH+4,SP       ;REMOVE STORAGE FROM STACK
                05  01FE            RSB                             ;RETURN STATUS
                    01FF
                    01FF   ;        .PAGE
```

```
                     01FF          .SBTTL  LNM$DELETE_HASH - DELETE ALL ENTRIES IN A HASH TABLE
                     01FF   ;+
                     01FF   ; LNM$DELETE_HASH - DELETE ALL ENTRIES IN A HASH TABLE
                     01FF   ;
                     01FF   ; THIS ROUTINE IS CALLED TO DELETE ALL NAMES WITH A SPECIFIED OR GREATER
                     01FF   ; ACCESS MODE FROM A SPECIFIED HASH TABLE.
                     01FF   ;
                     01FF   ; INPUTS:
                     01FF   ;
                     01FF   ;       R2 = ACCESS MODE.
                     01FF   ;       R3 = ADDRESS OF HASH TABLE.
                     01FF   ;       R4 = PCB ADDRESS.
                     01FF   ;
                     01FF   ;       THE LOGICAL NAME MUTEX WILL BE LOCKED FOR WRITE ACCESS,
                     01FF   ;       AND THE CALLER IS ASSUMED TO HAVE THE PRIVILEGE OF DELETING
                     01FF   ;       THE LOGICAL NAME TABLE ENTRIES WITHIN THE SPECIFIED TABLE.
                     01FF   ;
                     01FF   ; OUTPUTS:
                     01FF   ;
                     01FF   ;       THE APPROPRIATE HASH TABLE IS SCANNED AND ALL NAMES
                     01FF   ;       WITH AN ACCESS MODE GREATER THAN OR EQUAL TO
                     01FF   ;       THE SPECIFIED ACCESS MODE ARE DELETED.  QUOTA IS RETURNED.
                     01FF   ;
                     01FF   ;       R1, R2, AND R3 ARE MODIFIED.
                     01FF   ;-
                     01FF
                     01FF  LNM$DELETE_HASH::
                     01FF           SAVIPL                          ;SAVE CURRENT IPL ON STACK
  7E    54    7D     0202           MOVQ    R4,-(SP)                ;SAVE REGISTERS
      056A    30     0205           BSBW    LNM$LOCKW               ;LOCK LOGICAL NAME MUTEX FOR WRITING
        51    D4     0208           CLRL    R1                      ;NO SPECIFIC TABLE HEADER
      FEB7    30     020A           BSBW    DELETE_NAMES            ;DELETE THE NAMES WITHIN THE TABLE
  54    8E    7D     020D           MOVQ    (SP)+,R4                ;RESTORE REGISTERS
      0567    30     0210           BSBW    LNM$UNLOCK              ;UNLOCK THE LOGICAL NAME MUTEX
                     0213           ENBINT                          ;RESTORE IPL TO ITS VALUE ON ENTRY
  50    01    3C     0216           MOVZWL  #SS$_NORMAL,R0          ;SUCCESS ALWAYS
              05     0219           RSB                             ;RETURN STATUS
                     021A
                     021A  ;        .PAGE
```

```
                        021A         .SBTTL  LNM$DELETE_TAB  - DELETE ALL ENTRIES IN A LOGICAL NAME TABLE
                        021A   ;+
                        021A   ; LNM$DELETE_TAB - DELETE ALL ENTRIES IN A LOGICAL NAME TABLE
                        021A   ;
                        021A   ; THIS ROUTINE IS CALLED TO DELETE ALL NAMES WITH A SPECIFIED OR GREATER
                        021A   ; ACCESS MODE FROM A SPECIFIED LOGICAL NAME TABLE.
                        021A   ;
                        021A   ; INPUTS:
                        021A   ;
                        021A   ;       R1 = ADDRESS OF TABLE HEADER.
                        021A   ;       R2 = ACCESS MODE.
                        021A   ;
                        021A   ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR WRITE ACCESS,
                        021A   ;       AND THAT THE CALLER HAS THE PRIVILEGE OF DELETING THE LOGICAL NAME
                        021A   ;       TABLE ENTRIES WITHIN THE SPECIFIED TABLE.
                        021A   ;
                        021A   ; OUTPUTS:
                        021A   ;
                        021A   ;       THE APPROPRIATE HASH TABLE IS SCANNED AND ALL NAMES CONTAINED IN
                        021A   ;       THE SPECIFIED TABLE WITH AN ACCESS MODE GREATER THAN OR EQUAL TO
                        021A   ;       THE SPECIFIED ACCESS MODE ARE DELETED.  QUOTA IS RETURNED.
                        021A   ;
                        021A   ;       R1, R2, AND R3 ARE MODIFIED.
                        021A   ;-
                        021A
                        021A   LNM$DELETE_TAB::                                 ;DELETE LOGICAL NAME TABLE ENTRY
        7E   54     7D  021A         MOVQ    R4,-(SP)                       ;SAVE REGISTERS
  53     01  A1     D0  021D         MOVL    LNMTH$L_HASH(R1),R3            ;ADDRESS OF HASH TABLE
           FEA0     30  0221         BSBW    DELETE_NAMES                   ;DELETE THE NAMES WITHIN THE TABLE
        54   8E     7D  0224         MOVQ    (SP)+,R4                       ;RESTORE REGISTERS
        50   01     3C  0227         MOVZWL  #SS$_NORMAL,R0                 ;SUCCESS ALWAYS
                    05  022A         RSB                                    ;RETURN STATUS
                        022B
                        022B   ;       .PAGE
```

```
                               022B         .SBTTL  LNM$INIT_PROT   - INIT A LOGICAL NAME TABLE'S OBJECT RIGHTS BLOCK
                               022B  ;+
                               022B  ; LNM$INIT_PROT - INIT A LOGICAL NAME TABLE'S OBJECT RIGHTS BLOCK
                               022B  ;
                               022B  ; THIS ROUTINE IS CALLED TO INITIALIZE THE OBJECT RIGHTS BLOCK FOR A SHAREABLE
                               022B  ; LOGICAL NAME TABLE. IT IS ASSUMED THAT THIS ROUTINE IS NEVER CALLED FOR A
                               022B  ; PROCESS-PRIVATE LOGICAL NAME TABLE. CURRENTLY, ONLY SOGW PROTECTION AND UIC
                               022B  ; ARE ACTUALLY USED.   ACCESS MODE CHECKING IS HANDLED BY THE CALLING ROUTINE
                               022B  ; AND ALL OTHER SECURITY CHECKS ARE NYI.
                               022B  ;
                               022B  ; INPUTS:
                               022B  ;
                               022B  ;       R0 = ADDRESS OF STORAGE TO USE FOR ORB.
                               022B  ;       R1 = ADDRESS OF PARENT LOGICAL NAME TABLE LNMTH.
                               022B  ;       R2 = ADDRESS OF LOGICAL NAME TABLE LNMTH.
                               022B  ;       R4 = ADDRESS OF LOGICAL NAME TABLE LNMB.
                               022B  ;       R6 = ACCESS MASK TO ASSIGN TO THE TABLE.
                               022B  ;       4(SP) = CURRENT PCB ADDRESS.
                               022B  ;
                               022B  ; OUTPUTS:
                               022B  ;
                               022B  ;       R0, R3 DESTROYED.
                               022B  ;       ALL OTHER REGISTERS PRESERVED.
                               022B  ;-
                               022B
                               022B
                               022B  LNM$INIT_PROT::                                  ;INIT ORB FOR A LOGICAL NAME TABLE
               50    07    C0  022B         ADDL2   #^X07,R0                       ;ALGIN ORB ON A QUADWORD BOUNDRY
               50    07    CA  022E         BICL2   #^X07,R0
            05 A2    60    DE  0231         MOVAL   (R0),LNMTH$L_ORB(R2)           ;SAVE ADDRESS OF ORB IN TABLE HEADER
            53    04 AE    D0  0235         MOVL    4(SP),R3                       ;RETRIEVE PCB ADDRESS
          80    00BC C3    D0  0239         MOVL    PCB$L_UIC(R3),(R0)+            ;SET OWNER UIC
          80    FFFF 8F    3C  023E         MOVZWL  #^XFFFF,(R0)+                  ;INITIALIZE ACL MUTEX
  80    00490058 8F    D0  0243         MOVL    #<DYN$C_ORB @ 16+-             ;SET STRUCTURE TYPE IN FLAGS FIELD
                               024A                 ORB$C_LENGTH>,(R0)+            ;SET STRUCTURE SIZE
                     80    D4  024A         CLRL    (R0)+                          ;SPARE WORD AND REF COUNT NOT USED
                     80    7C  024C         CLRQ    (R0)+                          ;NO ACCESS MODE CHECKS ARE MADE
  80    56    04    00    EF  024E         EXTZV   #0, #4,R6,(R0)+                ;SET SYSTEM PROTECTION FIELD
  80    56    04    04    EF  0253         EXTZV   #4, #4,R6,(R0)+                ;SET OWNER PROTECTION FIELD
  80    56    04    08    EF  0258         EXTZV   #8, #4,R6,(R0)+                ;SET GROUP PROTECTION FIELD
  80    56    04    0C    EF  025D         EXTZV   #12,#4,R6,(R0)+                ;SET WORLD PROTECTION FIELD
                     80    7C  0262         CLRQ    (R0)+                          ;NOTE NO ACL AS YET
                     34    BB  0264         PUSHR   #^M<R2,R4,R5>                  ;SAVE REGISTERS (R1 PRESERVED BY MOVC)
          00    61    00    2C  0266         MOVC5   #0,(R1),#0,-                   ;INITIALIZE MINIMUM AND MAXIMUM
                60    28                            #<ORB$S_MIN_CLASS+-            ;CLASSIFICATION MASKS TO 0
                               026C                 ORB$S_MAX_CLASS>,(R0)
                     34    BA  026C         POPR    #^M<R2,R4,R5>                  ;RESTORE REGISTERS
                           05  026E         RSB                                    ;RETURN
                               026F
                               026F  ;      .PAGE
```

```
026F        .SBTTL  LNM$INSLOGTAB   - INSERT IN LOGICAL NAME TABLE BY ADDRESS
026F  ;+
026F  ; LNM$INSLOGTAB - INSERT IN LOGICAL NAME TABLE BY ADDRESS
026F  ;
026F  ; THIS ROUTINE IS CALLED TO INSERT A NEW ENTRY INTO THE LOGICAL NAME TABLE
026F  ; SPECIFIED BY TABLE HEADER ADDRESS. INSERTION IN THE CASE OF AN ENTRY FOR A
026F  ; NEW LOGICAL NAME TABLE INCLUDES THE LINKING OF THE NEW TABLE ENTRY TO PARENT
026F  ; AND SIBLINGS AND ANY REQUIRED QUOTA DEDUCTIONS.
026F  ;
026F  ; IF AN ENTRY (OR ENTRIES) ALREADY EXISTS AT AN OUTER ACCESS MODE AND THE NEW
026F  ; NAME IS UNALIASABLE, THE EXISTING NAME(S) IS (ARE) DELETED.
026F  ;
026F  ; IF AN UNALIASABLE ENTRY ALREADY EXISTS IN AN INNER ACCESS MODE, AN ERROR IS
026F  ; RETURNED.
026F  ;
026F  ; IF AN EQUIVALENT ENTRY ALREADY EXISTS, IT IS DELETED UNLESS CREATE_IF IS
026F  ; SPECIFIED IN WHICH CASE THE NEW ENTRY WILL BE JUST BE MAPPED TO THE EXISTING
026F  ; ENTRY, AND NO NAMES ARE SUPERSEDED.
026F  ;
026F  ; IF THE NEW ENTRY IS FOR A LOGICAL NAME TO BE CONTAINED WITHIN EITHER THE
026F  ; PROCESS OR SYSTEM DIRECTORY LOGICAL NAME TABLE, THEN THE HASH CODE VALUE OF
026F  ; EACH OF THE LOGICAL NAME'S EQUIVALENCE STRINGS IS COMPUTED AND STORED WITHIN
026F  ; THE CORRESPONDING TRANSLATION BLOCK.
026F  ;
026F  ; INPUTS:
026F  ;
026F  ;       R1 = ADDRESS OF LOGICAL NAME BLOCK.
026F  ;            (ADDRESS OF TABLE IS IN THE BLOCK).
026F  ;       R2 = ATTRIBUTES AFFECTING TABLE ENTRY CREATION.
026F  ;            (IT IS ASSUMED THAT THE HIGH ORDER BIT IS UNUSED AND 0).
026F  ;
026F  ;       IT IS ASSUMED THAT THERE IS SUFFICIENT QUOTA IN THE CONTAINING TABLE
026F  ;       (AND THE PARENT'S QUOTA HOLDER IN THE CASE OF A LOGICAL NAME TABLE
026F  ;       ENTRY) FOR THE INSERTION OF THE NEW ENTRY AND THE DEDUCTION OF ITS
026F  ;       SEPARATE QUOTA WHEN APPROPRIATE.
026F  ;
026F  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR WRITE ACCESS.
026F  ;
026F  ; OUTPUTS:
026F  ;
026F  ;       R0 CONTAINS A SUCCESS STATUS.
026F  ;
026F  ;               THE LOGICAL NAME IS INSERTED IN THE SPECIFIED TABLE.
026F  ;               QUOTA IS DEDUCTED WHEN APPROPRIATE.
026F  ;
026F  ;               R0 = SS$_LNMCREATED - NEW TABLE WAS INSERTED.
026F  ;               R0 = SS$_NORMAL     - NEW TABLE WAS MAPPED TO EXISTING TABLE OR
026F  ;                                     NEW LOGICAL NAME WAS INSERTED.
026F  ;               R0 = SS$_SUPERSEDE  - LOGICAL NAME SUPERCEDED EXISTING NAME.
026F  ;
026F  ;       R1 CONTAINS ADDRESS OF LNMB MAPPED TO IF CREATE-IF SET AND NEW LOGICAL
026F  ;               NAME TABLE ENTRY IS MAPPED TO EXISTING ENTRY.
026F  ;
026F  ;       R0 CONTAINS AN ERROR STATUS.
026F  ;
026F  ;               R0 = SS$_DUPLNAME   - NON-ALIASABLE DUPLICATE EXISTS.
026F  ;               R0 = SS$_PARENT_DEL - DELETEION OF PARENT WOULD HAVE OCCURRED.
```

```
                               026F  ;
                               026F  ;
                               026F  ;       R1, R2, R3, AND R5 ARE MODIFIED
                               026F  ;
                               026F  ;-
                               026F
                               026F  LNM$INSLOGTAB::                              ;INSERT IN LOGICAL NAME TABLE ENTRY
             55    51    D0    026F          MOVL    R1,R5                        ;ADDRESS OF NAME BLOCK
                               0272
                               0272  ;
                               0272  ; BUILD A NAME TRANSLATION CONTROL BLOCK FOR THE NEW TABLE ENTRY ON THE STACK.
                               0272  ;
                               0272
                   7E    D4    0272          CLRL    -(SP)                        ;ZERO NT_L_THREAD
                0C A5    DD    0274          PUSHL   LNMB$L_TABLE(R5)             ;ADDRESS OF CONTAINING TABLE HEADER
                12 A5    9F    0277          PUSHAB  LNMB$T_NAME+1(R5)            ;ADDRESS OF NAME
             7E 11 A5    9A    027A          MOVZBL  LNMB$T_NAME(R5),-(SP)        ;LENGTH OF NAME
          00000403 8F    DD    027E          PUSHL   #<NT_M_MODIFY @ 8+-          ;HASH BUCKET MIGHT BE MODIFIED
                               0284                  PSL$C_USER>                  ;SPECIFY ACCESS MODE AS USER
       04 10 A5    00    E0    0284          BBS     #LNMB$V_NO_ALIAS, -          ;BRANCH IF NOT ALIASABLE AND ALL ACCESS
                               0289                  LNMB$B_FLAGS(R5),10$         ;MODES WILL BE CONSIDERED
          6E    0B A5    90    0289          MOVB    LNMB$B_ACMODE(R5),(SP)       ;SKIP CONSIDERATION OF OUTER ACCESS MODE
             53    5E    D0    028D  10$:    MOVL    SP,R3                        ;ADDRESS OF TRANSLATION BLOCK
                               0290
                               0290  ;
                               0290  ; SEARCH FOR AN EXISTING LOGICAL NAME TABLE ENTRY WITH A NAME MATCHING THE NAME
                               0290  ; OF THE NEW TABLE ENTRY. THERE CAN BE TEN OUTCOMES OF SUCH A SEARCH AND THEY
                               0290  ; ARE LISTED BELOW:
                               0290  ;
                               0290  ; 1. AN EXISTING LOGICAL NAME TABLE ENTRY WAS NOT FOUND AT THE ACCESS MODE
                               0290  ;    SEARCHED.
                               0290  ;
                               0290  ;    A. THE CREATE_IF BIT IS SET:
                               0290  ;       IF IDENTICAL NAMES AT OUTER ACCESS MODE HAD BEEN SEEN, THEN THESE
                               0290  ;       LOGICAL NAME TABLE ENTRIES MUST BE DELETED BEFORE THE NEW ENTRY CAN BE
                               0290  ;       INSERTED. THIS IS DONE BY RE-STARTING FROM THE BEGINNING THE SEARCH FOR
                               0290  ;       AN EXISTING LOGICAL NAME TABLE ENTRY WITH A NAME MATCHING THE NAME OF
                               0290  ;       THE NEW TABLE ENTRY AFTER CLEARING THE CREATE_IF BIT.
                               0290  ;
                               0290  ;    B. THE CREATE_IF BIT IS NOT SET:
                               0290  ;       THE NEW LOGICAL NAME TABLE ENTRY IS INSERTED.
                               0290  ;
                               0290  ; 2. AN EXISTING LOGICAL NAME TABLE ENTRY IS FOUND BUT IT IS AT AN INNER ACCESS
                               0290  ;    MODE.
                               0290  ;
                               0290  ;    A. THE EXISTING LOGICAL NAME TABLE ENTRY DOES NOT ALLOW ALLIASES:
                               0290  ;       AN ERROR IS RETURNED.
                               0290  ;
                               0290  ;    B. THE CREATE_IF BIT IS SET AND THE EXISTING NAME ALLOWS ALLIASES:
                               0290  ;       IF IDENTICAL NAMES AT OUTER ACCESS MODE HAD BEEN SEEN, THEN THESE
                               0290  ;       LOGICAL NAME TABLE ENTRIES MUST BE DELETED BEFORE THE NEW ENTRY CAN BE
                               0290  ;       INSERTED. THIS IS DONE BY RE-STARTING FROM THE BEGINNING THE SEARCH FOR
                               0290  ;       AN EXISTING LOGICAL NAME TABLE ENTRY WITH A NAME MATCHING THE NAME OF
                               0290  ;       THE NEW TABLE AFTER CLEARING THE CREATE_IF BIT.
                               0290  ;
                               0290  ;    C. THE CREATE_IF BIT IS NOT SET AND THE EXISTING NAME ALLOWS ALLIASES:
                               0290  ;       THE NEW LOGICAL NAME TABLE ENTRY IS INSERTED.
```

```
                           0290   ;
                           0290   ; 3. AN EXISTING LOGICAL NAME TABLE ENTRY IS FOUND BUT IT IS AT AN OUTER ACCESS
                           0290   ;    MODE.
                           0290   ;
                           0290   ;    A. THE CREATE_IF BIT IS SET:
                           0290   ;       A RECORD IS KEPT THAT A LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AS
                           0290   ;       THE NEW TABLE ENTRY WAS SEEN, AND THE SEARCH CONTINUES.
                           0290   ;
                           0290   ;    B. THE CREATE_IF BIT IS NOT SET:
                           0290   ;       THE EXISTING LOGICAL NAME TABLE ENTRY IS DELETED, AND THE SEARCH
                           0290   ;       CONTINUES.
                           0290   ;
                           0290   ; 4. AN EXISTING LOGICAL NAME TABLE ENTRY IS FOUND AT THE SAME ACCESS MODE.
                           0290   ;
                           0290   ;    A. THE CREATE_IF BIT IS SET:
                           0290   ;       THE NEW ENTRY IS MAPPED TO THE EXISTING ENTRY WITH THE SAME NAME AND
                           0290   ;       ACCESS MODE.
                           0290   ;
                           0290   ;    B. THE CREATE_IF BIT IS NOT SET BUT THE NODELETE BIT IS:
                           0290   ;       AN ERROR IS RETURNED INFORMING THAT THE CALLER ATTEMPTED TO DELETE
                           0290   ;       THE PARENT TABLE SPECIFIED (EITHER DIRECTLY OR INDIRECTLY THROUGH THE
                           0290   ;       DELETION OF A GRANDPARENT).
                           0290   ;
                           0290   ;    C. NEITHER THE CREATE_IF BIT NOR THE NODELETE BIT IS SET:
                           0290   ;       THE EXISTING ENTRY IS DELETED, AND THE SEARCH CONTINUES.
                           0290   ;
                           0290   ;
                           0290
                52    DD   0290            PUSHL   R2                      ;SAVE THE ATTRIBUTES
                01    DD   0292            PUSHL   #SS$_NORMAL             ;ASSUME SUCCESS
              0234    30   0294   20$:     BSBW    LNM$PRESEARCH           ;SEARCH FOR NAME IN TABLE
                           0297
             4C 50    E9   0297   25$:     BLBC    R0,50$                  ;NOTHING FOUND
     08 A5   08 A1    91   029A            CMPB    LNMB$B_ACMODE(R1), -    ;COMPARE ACCESS MODES
                           029F                    LNMB$B_ACMODE(R5)
                3D    1F   029F            BLSSU   40$                     ;BRANCH IF OWNED BY AN INNER MODE
                10    1A   02A1            BGTRU   30$                     ;BRANCH IF OWNED BY AN OUTER MODE
                           02A3
                           02A3   ;
                           02A3   ; AN EXISTING LOGICAL NAME TABLE ENTRY AT THE SAME ACCESS MODE WAS FOUND TO
                           02A3   ; HAVE THE SAME NAME AS THAT OF THE NEW TABLE ENTRY. EITHER MAP THE NEW ENTRY
                           02A3   ; TO THE EXISTING ENTRY, OR DELETE THE EXISTING ENTRY DEPENDING UPON THE
                           02A3   ; SETTING OF CREATE_IF.
                           02A3   ;
                           02A3
  03 04 AE      18    E1   02A3            BBC     #LNM$V_CREATE_IF,4(SP),27$ ;GO RETURN IF MAPPING NEW TABLE ENTRY
              00E5    31   02A8            BRW     90$                     ;ONTO EXISTING TABLE ENTRY
                04    E1   02AB   27$:     BBC     #LNMB$V_NODELETE,-      ;IF TABLE ENTRY IS A (GRAND)PARENT TO
          03 10 A1         02AD                    LNMB$B_FLAGS(R1),30$    ;THE NEW TABLE ENTRY THEN RETURN ERROR
              00EB    31   02B0            BRW     110$
                           02B3
                           02B3   ;
                           02B3   ; AN EXISTING LOGICAL NAME TABLE ENTRY AT AN OUTER ACCESS MODE WAS FOUND TO
                           02B3   ; HAVE THE SAME NAME AS THAT OF THE NEW TABLE ENTRY. EITHER REMEMBER THAT SUCH
                           02B3   ; A LOGICAL NAME TABLE ENTRY HAS BEEN ENCOUNTERED, OR DELETE IT DEPENDING UPON
                           02B3   ; THE SETTING OF CREATE_IF.
                           02B3   ;
```

```
                               02B3
0B 04 AE    18    E1   02B3  30$:    BBC     #LNM$V_CREATE_IF,4(SP),35$ ;BRANCH IF CREATE_IF NOT SET
00 04 AE    1F    E2   02B8          BBSS    #31,4(SP),31$              ;MARK OUTER ACCESS NAMES SEEN
   10 A3    61    9E   02BD  31$:    MOVAB   (R1),NT_L_THREAD(R3)       ;SETUP TO CONTINUE WITH NEXT BLOCK
            16    11   02C1          BRB     37$                        ;AND GO CONTINUE SEARCH
                       02C3
  6E    0631 8F   3C   02C3  35$:    MOVZWL  #SS$_SUPERSEDE,(SP)        ;UPDATE STATUS RETURN
            53    DD   02C8          PUSHL   R3                         ;SAVE REGISTERS OVER DELETE
          FED9    30   02CA          BSBW    LNM$DELETE                 ;DELETE A LOGICAL NAME BLOCK
            53  8ED0   02CD          POPL    R3                         ;RESTORE THINGS
         06 50    E8   02D0          BLBS    R0,37$                     ;CONTINUE IF DELETION SUCCESSFUL
      6E    50    D0   02D3          MOVL    R0,(SP)                    ;OTHERWISE SAVE ERROR AND TERMINATE
          00B7    31   02D6          BRW     90$                        ;TABLE ENTRY INSERTION ATTEMPT
                       02D9
          0219    30   02D9  37$:    BSBW    LNM$CONTSEARCH             ;CONTINUE SEARCHING FOR MORE MODES
            B9    11   02DC          BRB     25$                        ;LOOP OVER ALL ALIASING NAMES
                       02DE
                       02DE  ;
                       02DE  ; AN EXISTING LOGICAL NAME TABLE ENTRY AT AN INNER ACCESS MODE WAS FOUND TO
                       02DE  ; HAVE THE SAME NAME AS THAT OF THE NEW TABLE ENTRY. IF THIS LOGICAL NAME TABLE
                       02DE  ; ENTRY DOES NOT ALLOW ALLIASES THEN RETURN AN ERROR. OTHERWISE, EITHER INSERT
                       02DE  ; THE NEW TABLE ENTRY, OR RE-START THE SEARCH FROM THE BEGINNING DEPENDING UPON
                       02DE  ; WHETHER CREATE_IF IS SET, AND IF IT IS, WHETHER ANY IDENTICAL NAMES WERE
                       02DE  ; FOUND TO EXIST AT OUTER ACCESS MODES.
                       02DE  ;
                       02DE
03 10 A1    00    E1   02DE  40$:    BBC     #LNMB$V_NO_ALIAS, -        ;CONTINUE IF ALIASABLE
                       02E3                  LNMB$B_FLAGS(R1),50$
          00B1    31   02E3          BRW     100$                       ;BRANCH IF NOT ALIASABLE
                       02E6
                       02E6  ;
                       02E6  ; AN EXISTING LOGICAL NAME TABLE ENTRY WAS NOT FOUND TO HAVE THE SAME SAME NAME
                       02E6  ; AS THAT OF THE NEW TABLE ENTRY AT THE ACCESS MODE SEARCHED. EITHER INSERT THE
                       02E6  ; NEW TABLE ENTRY, OR RE-START THE SEARCH FROM THE BEGINNING DEPENDING UPON
                       02E6  ; WHETHER CREATE_IF IS SET, AND IF IT IS, WHETHER ANY IDENTICAL NAMES WERE
                       02E6  ; FOUND TO EXIST AT OUTER ACCESS MODES.
                       02E6  ;
                       02E6
0A 04 AE    18    E5   02E6  50$:    BBCC    #LNM$V_CREATE_IF,4(SP),55$;INSERT NEW ENTRY IF CREATE_IF NOT SET
05 04 AE    1F    E5   02EB          BBCC    #31,4(SP),55$              ;OR IF OUTER ACCESS MODE NAMES NOT SEEN
         10 A3    D4   02F0          CLRL    NT_L_THREAD(R3)            ;GET SET TO RE-START SEARCH
            9F    11   02F3          BRB     20$                        ;GO RE-START SEARCH
                       02F5
   52    10 A3    D0   02F5  55$:    MOVL    NT_L_THREAD(R3),R2         ;ADDRESS OF PREVIOUS LNMB BLOCK
   04 A5    52    D0   02F9          MOVL    R2,LNMB$L_BLINK(R5)        ;SET NEW LNMB BLOCKS BACKPOINTER
      62    55    D0   02FD          MOVL    R5,LNMB$L_FLINK(R2)        ;RESET PREVIOUS LNMB BLOCKS FRONTPOINTER
      65    51    D0   0300          MOVL    R1,LNMB$L_FLINK(R5)        ;SET NEW LNMB BLOCKS FRONTPOINTER
            04    13   0303          BEQL    60$                        ;BRANCH IF NEW LNMB BLOCK IS AT LIST END
   04 A1    55    D0   0305          MOVL    R5,LNMB$L_BLINK(R1)        ;RESET NEXT LNMB BLOCKS BACKPOINTER
                       0309
                       0309  ;
                       0309  ; IF THE NEW ENTRY IS A LOGICAL NAME TABLE THEN THE FOLLOWING ACTIONS ARE
                       0309  ; TAKEN:
                       0309  ;
                       0309  ; 1. THE LOGICAL NAME BLOCK IS LINKED IN AS THE IMMEDIATE CHILD OF ITS PARENT
                       0309  ;    TABLE, AND THUS, AS THE FIRST SIBLING IN THE LIST OF SIBLINGS.
                       0309  ;
```

```
                    0309  ; 2. THE STATUS OF THE INSERTION IS CHANGED TO SS$_LNMCREATED IF NO LOGICAL NAME
                    0309  ;    TABLE ENTRIES HAD BEEN DELETED.
                    0309  ;
                    0309  ; 3. QUOTA CONSISTING OF ANY QUOTA SPECIFICALLY ALLOCATED TO THE NEW TABLE IS
                    0309  ;    DEDUCTED FROM THE QUOTA HOLDER OF THE PARENT LOGICAL NAME TABLE.
                    0309  ;
                    0309
           03   E1  0309  60$:    BBC     #LNMB$V_TABLE,-          ;BRANCH IF THE NAME BLOCK IS NOT FOR
     2F 10 A5       030B                  LNMB$B_FLAGS(R5),80$     ;A NEW LOGICAL NAME TABLE
                    030E
  50   11 A5    9A  030E  70$:    MOVZBL  LNMB$T_NAME(R5),R0       ;SIZE OF NAME STRING
52    12 A540   9E  0312          MOVAB   LNMB$T_NAME+1(R5)[R0],R2;ADDRESS OF TRANSLATION BLOCK
  52   05 A2    9E  0317          MOVAB   LNMX$T_XLATION+1(R2),R2 ;ADDRESS OF BLOCKS TABLE HEADER
                    031B
  51   0D A2    D0  031B          MOVL    LNMTH$L_PARENT(R2),R1    ;ADDRESS OF PARENT'S TABLE HEADER
       11 A1    D0  031F          MOVL    LNMTH$L_CHILD(R1),-      ;LINK IN NEW TABLE ENTRY AS THE
       15 A2        0322                  LNMTH$L_SIBLING(R2)      ;IMMEDIATE CHILD OF THE PARENT AND
  11 A1   62    9E  0324          MOVAB   (R2),LNMTH$L_CHILD(R1)   ;AS THE FIRST SIBLING IN THE LIST
                    0328
6E    0631 8F   B1  0328          CMPW    #SS$_SUPERSEDE,(SP)      ;WAS A LOGIGAL NAME SUPERSEDED?
           05   13  032D          BEQL    75$                      ;BRANCH IF YES
6E    06B1 8F   3C  032F          MOVZWL  #SS$_LNMCREATED,(SP)     ;CHANGE STATUS IF NO
                    0334
  51   19 A1    D0  0334  75$:    MOVL    LNMTH$L_QTABLE(R1),R1    ;RETRIEVE PARENT'S QUOTA HOLDER
       1D A2    C2  0338          SUBL2   LNMTH$L_BYTESLM(R2),-    ;SUBTRACT QUOTA TO BE SPECIFICALLY
       21 A1        033B                  LNMTH$L_BYTES(R1)        ;ALLOCATED TO THE NEW TABLE
                    033D
                    033D  ;
                    033D  ; DEDUCT THE SIZE OF THE NEW LOGICAL NAME ENTRY FROM THE QUOTA HOLDER OF THE
                    033D  ; CONTAINING LOGICAL NAME TABLE.
                    033D  ;
                    033D
  53   08 A5    3C  033D  80$:    MOVZWL  LNMB$W_SIZE(R5),R3       ;RETRIEVE SIZE OF NEW LOGICAL NAME ENTRY
  51   0C A5    D0  0341          MOVL    LNMB$L_TABLE(R5),R1      ;RETRIEVE CONTAINING TABLE HEADER ADDR
  52   19 A1    D0  0345          MOVL    LNMTH$L_QTABLE(R1),R2    ;RETRIEVE QUOTA HOLDER'S ADDRESS
  21 A2   53    C2  0349          SUBL2   R3,LNMTH$L_BYTES(R2)     ;SUBTRACT SIZE OF NEW TABLE ENTRY
                    034D
                    034D  ;
                    034D  ; BUMP THE APPROPRIATE DIRECTORY SEQUENCE NUMBER IF THE CONTAINING TABLE
                    034D  ; IS ONE OF THE DIRECTORY TABLES.
                    034D  ;
                    034D
           01   E1  034D          BBC     #LNMTH$V_DIRECTORY,-     ;DIRECTORY TABLE?
        3F 61       034F                  LNMTH$B_FLAGS(R1),90$    ;OKAY IF NOT
  08 51    1F   E0  0351          BBS     #31,R1,82$               ;BRANCH IF SYSTEM DIRECTORY
 00000000'9F    D6  0355          INCL    @#CTL$GL_LNMDIRSEQ       ;BUMP PROCESS DIRECTORY COUNTER
           06   11  035B          BRB     83$
 00000000'9F    D6  035D  82$:    INCL    @#LNM$GL_SYSDIRSEQ       ;BUMP SYSTEM DIRECTORY COUNTER
                    0363
                    0363  ;
                    0363  ; IF THE NEW ENTRY IS FOR A LOGICAL NAME TO BE CONTAINED WITHIN EITHER THE
                    0363  ; PROCESS OR SYSTEM DIRECTORY LOGICAL NAME TABLE, THEN THE HASH CODE VALUE OF
                    0363  ; EACH OF THE LOGICAL NAME'S EQUIVALENCE STRINGS IS COMPUTED AND STORED WITHIN
                    0363  ; THE CORRESPONDING TRANSLATION BLOCK.
                    0363  ;
                    0363
           03   E0  0363  83$:    BBS     #LNMB$V_TABLE,-          ;SKIP COMPUTATION AND STORAGE OF HASH
```

```
          28 10 A5       0365                      LNMB$B_FLAGS(R5),90$     ;CODES IF THIS IS A LOGICAL NAME TABLE
                         0368
   55    11 A5     9E    0368             MOVAB    LNMB$T_NAME(R5),R5       ;RETRIEVE ADDRESS AND SIZE OF LOGICAL
      50    85     9A    036C             MOVZBL   (R5)+,R0                 ;NAME'S NAME
      55    50     C0    036F             ADDL2    R0,R5                    ;POSITION TO FIRST TRANSLATION BLOCK
                         0372
            02     E0    0372   85$:      BBS      #LNMX$V_XEND,-           ;GO RETURN IF LAST TRANSLATION BLOCK
         1A 65           0374                      LNMX$B_FLAGS(R5),90$
   51    04 A5     9E    0376             MOVAB    LNMX$T_XLATION(R5),R1    ;RETRIEVE ADDRESS AND SIZE OF CURRENT
      50    81     9A    037A             MOVZBL   (R1)+,R0                 ;TRANSLATION BLOCK'S EQUIVALENCE STRING
      7E    50     7D    037D             MOVQ     R0,-(SP)                 ;SAVE ADDRESS AND SIZE ON STACK
          01EB     30    0380             BSBW     LNM$HASH                 ;DETERMINE AND STORE THE CURRENT
   02 A5    50     B0    0383             MOVW     R0,LNMX$W_HASH(R5)       ;EQUIVALENCE STRING'S HASH CODE VALUE
      50    8E     7D    0387             MOVQ     (SP)+,R0                 ;RESTORE ADDRESS AND SIZE OF STRING
 55   51    50     C1    038A             ADDL3    R0,R1,R5                 ;POSITION TO NEXT TRANSLATION BLOCK
            E2     11    038E             BRB      85$                      ;AND CONTINUE
                         0390
                         0390       ;
                         0390       ; RETRIEVE THE STATUS TO BE RETURNED, CLEAN THE NAME TRANSLATION CONTROL BLOCK
                         0390       ; FROM THE STACK AND RETURN.
                         0390       ;
                         0390
            50 8ED0      0390   90$:      POPL     R0                       ;FETCH STATUS
      5E    18     C0    0393             ADDL     #NT_K_LENGTH+4,SP        ;CLEAN BLOCK FROM STACK
                   05    0396             RSB                               ;EXIT
 6E    0094 8F     3C    0397   100$:     MOVZWL   #SS$_DUPLNAM,(SP)        ;TRIED TO SUPERCEDE UNALIASABLE NAME
            F2     11    039C             BRB      90$                      ;JOIN MAIN EXIT
 6E    2254 8F     3C    039E   110$:     MOVZWL   #SS$_PARENT_DEL,(SP)     ;TRIED TO DELETE (GRAND)PARENT
            EB     11    03A3             BRB      90$                      ;JOIN MAIN EXIT
                         03A5
                         03A5       ;         .PAGE
                         03A5          .SBTTL  LNM$SEARCHLOG   - SEARCH FOR LOGICAL NAME
                         03A5    ;+
                         03A5    ; LNM$SEARCHLOG - SEARCH FOR LOGICAL NAME
                         03A5    ;
                         03A5    ; THIS ROUTINE IS CALLED TO SEARCH FOR A LOGICAL NAME MATCH IN A LIST OF
                         03A5    ; LOGICAL NAME TABLES.
                         03A5    ;
                         03A5    ; INPUTS:
                         03A5    ;
                         03A5    ;       R0 = LENGTH OF LOGICAL NAME STRING.
                         03A5    ;       R1 = ADDRESS OF LOGICAL NAME STRING.
                         03A5    ;       R2 = LENGTH OF TABLE NAME STRING.
                         03A5    ;       R3 = ADDRESS OF TABLE NAME STRING.
                         03A5    ;       R5 = SEARCH ACCESS MODE IN LOW BYTE,
                         03A5    ;            CASELESS FLAG IN BIT 8,
                         03A5    ;            HIGH ORDER WORD 0.
                         03A5    ;
                         03A5    ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR READ ACCESS.
                         03A5    ;
                         03A5    ; OUTPUTS:
                         03A5    ;
                         03A5    ;       R0 LOW BIT CLEAR INDICATES SEARCH FAILURE.
                         03A5    ;
                         03A5    ;               R0 = SS$_NOLOGNAM - NO LOGICAL NAME MATCH FOUND.
                         03A5    ;               R1 = ADDRESS OF LOGICAL NAME BLOCK ON WHICH SEARCH FAILED.
                         03A5    ;
                         03A5    ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                         03A5    ;
                         03A5    ;               R1 = ADDRESS OF LOGICAL NAME BLOCK THAT CONTAINS MATCH.
                         03A5    ;
                         03A5    ;       ALL OTHER REGISTERS ARE PRESERVED.
                         03A5    ;-
                         03A5
                         03A5   LNM$SEARCHLOG::                                 ;SEARCH FOR LOGICAL NAME
              55     DD  03A5          PUSHL   R5                              ;SAVE REGISTERS
         7E   52     7D  03A7          MOVQ    R2,-(SP)
                         03AA
                         03AA    ;
                         03AA    ; PERFORM A PRE-SEARCH TO SEE IF THE TARGET LOGICAL NAME EXISTS AT ALL
                         03AA    ; INDEPENDANT OF EITHER CONTAINING TABLE HEADER ADDRESS OR ACCESS MODE. IF
                         03AA    ; THE NAME EXISTS WITHIN THE PROCESS-PRIVATE NAME SPACE WE WILL ALSO BE
                         03AA    ; PRE-POSITIONED TO THE FIRST LNMB IN THE LINKED LIST OF PROCES-PRIVATE LNMBs
                         03AA    ; WITH THIS TARGET NAME. LIKEWISE, IF THE NAME EXISTS WITHIN THE SHAREABLE NAME
                         03AA    ; SPACE WE WILL ALSO BE PRE-POSITIONED TO THE FIRST SHAREABLE LNMB IN THE LINKED
                         03AA    ; LIST OF LNMBS WITH THIS TARGET NAME.
                         03AA    ;
                         03AA
              7E     7C  03AA          CLRQ    -(SP)                           ;NAME BLOCK ADDRESS AND TABLE ID
         7E   50     7D  03AC          MOVQ    R0,-(SP)                        ;TABLE NAME AND ADDRESS
              55     DD  03AF          PUSHL   R5                              ;ACCESS MODE AND CASE FLAG
         53   5E     D0  03B1          MOVL    SP,R3                           ;ADDRESS OF NAME TRANSLATION BLOCK
 55   00000000'EF    DE  03B4          MOVAL   L^LNM$AL_HASHTBL,R5             ;ADDRESS OF TABLE ADDRESS POINTERS
      0F 5E   1F     E0  03BB          BBS     #31,SP,10$                      ;BRANCH IF SMALL PROCESS
      51   04 B5     D0  03BF          MOVL    @4(R5),R1                       ;ADDRESS OF PROCESS HASH TABLE
              09     13  03C3          BEQL    10$                             ;SKIP IF NO TABLE DEFINED
                         03C5
            0103     30  03C5          BSBW    LNM$PRESEARCH                   ;FIND HEAD OF POSSIBLE TRANSLATIONS
      03 50   E8  03C8          BLBS    R0,10$                  ;BRANCH IF NO POSSIBLE TRANSLATION
      10 A3   D4  03CB          CLRL    NT_L_THREAD(R3)         ;CLEAR NAME BLOCK ADDRESS
                  03CE
         7E   7C  03CE  10$:    CLRQ    -(SP)                   ;MAKE ANOTHER NAME TRANSLATION BLOCK
   7E  04 A3  7D  03D0          MOVQ    NT_L_NAMLEN(R3),-(SP)   ;TABLE NAME AND ADDRESS
         63   DD  03D4          PUSHL   (R3)                    ;CONTROL LONGWORD
      53 5E   D0  03D6          MOVL    SP,R3                   ;ADDRESS OF SYSTEM NAME TRANSLATION BLOCK
      51 95   D0  03D9          MOVL    @(R5)+,R1               ;ADDRESS OF SYSTEM HASH TABLE
                  03DC
       00EC   30  03DC          BSBW    LNM$PRESEARCH           ;FIND HEAD OF POSSIBLE TRANSLATIONS
      08 50   E8  03DF          BLBS    R0,20$                  ;BRANCH IF TRANSLATION MAY EXIST
      10 A3   D4  03E2          CLRL    NT_L_THREAD(R3)         ;NO POSSIBLE TRANSLATION
                  03E5
                  03E5  ;
                  03E5  ; IF THE SPECIFIED LOGICAL NAME DOES NOT EXIST IN EITHER THE PROCESS-PRIVATE OR
                  03E5  ; SHAREABLE NAME SPACES, REGARDLESS OF CONTAINING TABLE HEADER ADDRESS OR ACCESS
                  03E5  ; MODE, THEN IT IS POINTLESS TO CONTINUE THE SEARCH, SO RETURN AN ERROR.
                  03E5  ; OTHERWISE, POSITION TO THE FIRST TARGET LOGICAL NAME TABLE IN WHICH TO SEARCH
                  03E5  ; FOR THE SPECIFIED LOGICAL NAME.
                  03E5  ;
                  03E5
      24 A3   D5  03E5          TSTL    NT_K_LENGTH+NT_L_THREAD(R3)     ;ANY POSSIBLE TRANSLATION?
         44   13  03E8          BEQL    90$                     ;BRANCH IF NO TRANSLATION POSSIBLE
                  03EA  20$:
   5E    2C   C2  03EA          SUBL    #RT_K_LENGTH-4,SP       ;ALLOCATE RECURSIVE TABLE NAME CONTROL BLOCK
         63   DD  03ED          PUSHL   NT_B_ACMODE(R3)         ;ACMODE, CASE FLAG
   55    5E   D0  03EF          MOVL    SP,R5                   ;ADDRESS OF BLOCK
52    28 A3   7D  03F2          MOVQ    2*NT_K_LENGTH(R3),R2    ;GET LOGICAL NAME TABLE DESCRIPTOR
                  03F6
       01F7   30  03F6          BSBW    LNM$SETUP               ;SETUP TABLE PROCESSING
      27 50   E9  03F9          BLBC    R0,70$                  ;NO TABLE FOUND
                  03FC
                  03FC  ;
                  03FC  ; SEARCH FOR THE SPECIFIED LOGICAL NAME WITHIN THE CURRENT LOGICAL NAME TABLE.
                  03FC  ; ONLY ONE NAME SPACE WILL HAVE TO BE SEARCHED FOR THE LOGICAL NAME, AND THAT
                  03FC  ; IS THE NAME SPACE THAT CORRESPONDS TO THE NAME SPACE THE CURRENT TABLE RESIDES
                  03FC  ; IN.
                  03FC  ;
                  03FC
53    30 A5   9E  03FC  40$:    MOVAB   RT_K_LENGTH(R5),R3      ;SYSTEM TABLE CONTROL BLOCK
03 51    1F   E0  0400          BBS     #31,R1,50$              ;BRANCH IF SYSTEM SPACE TABLE
   53    14   C0  0404          ADDL    #NT_K_LENGTH,R3         ;ADVANCE TO PROCESS TABLE CONTROL BLOCK
      10 A3   D5  0407  50$:    TSTL    NT_L_THREAD(R3)         ;ANY NAMES TO BE LOOKED AT?
         11   13  040A          BEQL    60$                     ;NO  - GO GET NEXT TABLE TO PROCESS
0C A3    51   D0  040C          MOVL    R1,NT_L_TABID(R3)       ;TABLE HEADER ADDRESS
                  0410
      10 A3   DD  0410          PUSHL   NT_L_THREAD(R3)         ;SAVE SEARCH CONTEXT
       00DF   30  0413          BSBW    LNM$CONTSEARCH          ;RESUME SEARCH FOR NAME
10 A3    8E   D0  0416          MOVL    (SP)+,NT_L_THREAD(R3)   ;RESTORE ORIGINAL SEARCH CONTEXT
      06 50   E8  041A          BLBS    R0,70$                  ;BRANCH IF NAME FOUND
                  041D
                  041D  ;
                  041D  ; THE SPECIFIED LOGICAL NAME WAS NOT FOUND WITHIN THE CURRENT LOGICAL NAME
                  041D  ; TABLE. POSITION TO THE NEXT TABLE IN THE SEARCH LIST OF LOGICAL NAME TABLES.
                  041D  ; IF THERE IS A NEXT TABLE, SEARCH FOR THE LOGICAL NAME WITHIN IT; OTHERWISE,
                  041D  ; THE SEARCH FOR THE SPECIFIED LOGICAL NAME IS TERMINATED WITH AN ERROR.
                  041D  ;
```

```
                       041D
          01F6    30   041D  60$:    BSBW    LNM$TABLE                    ;CONTINUE TABLE PROCESSING
         D9 50    E8   0420          BLBS    R0,40$                       ;BRANCH TO PROCESS ANOTHER TABLE
                       0423
   5E     58 AE   9E   0423  70$:    MOVAB   <2*NT_K_LENGTH>+RT_K_LENGTH(SP),SP     ;REMOVE TABLE SEARCH CONTROL
      52     8E   7D   0427  80$:    MOVQ    (SP)+,R2                     ;RESTORE REGISTERS
             55 8ED0   042A          POPL    R5
                  05   042D          RSB
  50    01BC 8F   3C   042E  90$:    MOVZWL  #SS$_NOLOGNAM,R0             ;NO TRANSLATION FOR NAME
        5E   28   C0   0433          ADDL2   #2*NT_K_LENGTH,SP            ;REMOVE NAME SEARCH CONTROL BLOCK
             EF   11   0436          BRB     80$                          ;JOIN MAIN EXIT
                       0438     ;    .PAGE
```

```
                             0438          .SBTTL  LNM$SEARCH_ONE  - SEARCH FOR LOGICAL NAME AND RETURN TRANSLATION
                             0438   ;+
                             0438   ; LNM$SEARCH_ONE - SEARCH FOR LOGICAL NAME AND RETURN INDEX 0 TRANSLATION
                             0438   ;
                             0438   ; THIS ROUTINE IS CALLED TO SEARCH FOR A LOGICAL NAME MATCH IN A LIST OF
                             0438   ; LOGICAL NAME TABLES. IF IT FINDS ONE, AND IF THAT LOGICAL NAME HAS A
                             0438   ; TRANSLATION WITH INDEX 0, THEN A COPY OF THE LNMX TRANSLATION BLOCK IS
                             0438   ; RETURNED IN THE SPECIFIED OUTPUT BUFFER.
                             0438   ;
                             0438   ; THIS ROUTINE DOES NOT PERFORM ANY ARGUEMENT VERIFICATION. HOWEVER, THIS
                             0438   ; ROUTINE DOES PERFORM ALL REQUIRED MUTEX LOCKING AND PROTECTION CHECKING.
                             0438   ;
                             0438   ; INPUTS:
                             0438   ;
                             0438   ;       R0 = LENGTH OF LOGICAL NAME STRING.
                             0438   ;       R1 = ADDRESS OF LOGICAL NAME STRING.
                             0438   ;       R2 = LENGTH OF TABLE NAME STRING.
                             0438   ;       R3 = ADDRESS OF TABLE NAME STRING.
                             0438   ;       R4 = PCB ADDRESS
                             0438   ;       R5 = SEARCH ACCESS MODE IN LOW BYTE, CASELESS FLAG IN BIT 8.
                             0438   ;       R6 = ADDRESS OF OUTPUT BUFFER
                             0438   ;            (MUST BE LNM$C_NAMLENGTH + LNMX$T_XLATION BYTES IN SIZE).
                             0438   ;
                             0438   ;
                             0438   ; OUTPUTS:
                             0438   ;
                             0438   ;       R0 LOW BIT CLEAR INDICATES SEARCH FAILURE.
                             0438   ;
                             0438   ;               R0 = SS$_NOLOGNAM - NO LOGICAL NAME MATCH FOUND.
                             0438   ;                                 - LOGICAL NAME FOUND BUT TRANSLATION WITH
                             0438   ;                                   INDEX 0 DOES NOT EXIST.
                             0438   ;
                             0438   ;               R0 = SS$_NOPRIV   - LOGICAL NAME WAS FOUND BUT CALLER DOES NOT
                             0438   ;                                   HAVE ACCESS TO THE SPECIFIED TABLE.
                             0438   ;
                             0438   ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                             0438   ;
                             0438   ;               A COPY OF THE LNMX FOR TRANSLATION INDEX 0 IN THE OUTPUT BUFFER.
                             0438   ;
                             0438   ;       REGISTERS R1 - R3 AND R5 ARE DESTROYED.
                             0438   ;-
                             0438
                             0438   LNM$SEARCH_ONE::                                ;SEARCH FOR LOGICAL NAME AND RETURN LNMX
              01      DD     0438           PUSHL   #SS$_NORMAL                     ;ASSUME SUCCESS
                             043A           SAVIPL                                  ;SAVE CURRENT IPL ON STACK
              50      DD     043D           PUSHL   R0                              ;SAVE LOGNAM STRING LENGTH
            0328      30     043F           BSBW    LNM$LOCKR                       ;LOCK LOGICAL NAME MUTEX FOR WRITING
              50    8ED0     0442           POPL    R0                              ;RESTORE LOGNAM STRING LENGTH
                             0445
        00000002             0445           .IF NE CA$_MEASURE                      ;CHECK FOR MEASUREMENT ENABLED
     00000000'EF      D6     0445           INCL    L^PMS$GL_LOGNAM                 ;IF YES COUNT CURRENT TRANSLATION
                             044B           .ENDC
                             044B
            FF57      30     044B           BSBW    LNM$SEARCHLOG                   ;SEARCH FOR THE LOGICAL NAME
           58 50      E9     044E           BLBC    R0,40$                          ;EXIT ON ANY ERROR
                             0451
        12 51   1F    E1     0451           BBC     #31,R1,1$                       ;ONLY CHECK ACCESS TO SHAREABLE TABLES
```

```
              51     DD  0455              PUSHL   R1                          ;SAVE LNMB ADDRESS
     51    0C A1     D0  0457              MOVL    LNMB$L_TABLE(R1),R1         ;RETRIEVE TABLE HEADER ADDRESS
        52    01     D0  045B              MOVL    #ARM$M_READ,R2              ;READ ACCESS
            FCD0     30  045E              BSBW    LNM$CHECK_PROT              ;PERFORM PROTECTION CHECK
          51 8ED0        0461              POPL    R1                          ;RESTORE LNMB ADDRESS
           42 50     E9  0464              BLBC    R0,40$                      ;EXIT ON ANY ERROR
                         0467
     51    11 A1     9E  0467      1$:     MOVAB   LNMB$T_NAME(R1),R1          ;ADDRESS OF NAME STRING
        50    81     9A  046B              MOVZBL  (R1)+,R0                    ;RETRIEVE SIZE OF NAME STRING
        51    50     C0  046E              ADDL2   R0,R1                       ;POSITION TO FIRST LNMX
                         0471
              02     E0  0471      5$:     BBS     #LNMX$V_XEND,-              ;IS THIS THE LAST TRANSLATION?
           12 61         0473                      LNMX$B_FLAGS(R1),10$        ;IF SO, NO INDEX 0 LNMX SO RETURN ERROR
           01 A1     95  0475              TSTB    LNMX$B_INDEX(R1)            ;IS THE INDEX 0 LNMX?
              15     13  0478              BEQL    20$                         ;IF SO, GO RETURN LNMX
              0B     14  047A              BGTR    10$                         ;IF POSITIVE INDEX THEN GO RETURN ERROR
     50    04 A1     9A  047C              MOVZBL  LNMX$T_XLATION(R1),R0       ;ELSE RETRIEVE SIZE OF TRANSLATION
 51    05 A140       9E  0480              MOVAB   LNMX$T_XLATION+1(R1)[R0],R1 ;POSITION TO NEXT LNMX
              EA     11  0485              BRB     5$                          ;GO SEE IF ITS INDEX IS 0
                         0487
04 AE    01BC 8F     3C  0487      10$:    MOVZWL  #SS$_NOLOGNAM,4(SP)         ;ELSE, RETURN THE APPROPRIATE ERROR
              10     11  048D              BRB     30$                         ;STATUS
                         048F
     50    04 A1     9A  048F      20$:    MOVZBL  LNMX$T_XLATION(R1),R0       ;RETRIEVE SIZE OF TRANSLATION STRING
        50    05     C0  0493              ADDL2   #LNMX$T_XLATION+1,R0        ;ADD SIZE OF LNMX OVERHEAD + COUNT FIELD
              54     DD  0496              PUSHL   R4                          ;SAVE PCB ADDRESS
   66    61   50     28  0498              MOVC3   R0,(R1),(R6)                ;MOVE ENTIRE LNMX FOR INDEX 0
          54 8ED0        049C              POPL    R4                          ;RESTORE PCB ADDRESS
                         049F
            02D8     30  049F      30$:    BSBW    LNM$UNLOCK                  ;UNLOCK THE LOGICAL NAME MUTEX
                         04A2              ENBINT                              ;RESTORE IPL TO ITS VALUE ON ENTRY
          50 8ED0        04A5              POPL    R0                          ;RESTORE STATUS
              05         04A8              RSB                                 ;RETURN
   04 AE      50     D0  04A9      40$:    MOVL    R0,4(SP)                    ;CHANGE RETURN STATUS TO AN ERROR STATUS
              F0     11  04AD              BRB     30$                         ;GO RETURN
                         04AF      ;       .PAGE
```

```
                        04AF          .SBTTL  LNM$FIRSTTAB    - SEARCH FOR FIRST TABLE NAME
                        04AF  ;+
                        04AF  ; LNM$FIRSTTAB - SEARCH FOR FIRST TABLE NAME
                        04AF  ;
                        04AF  ; THIS ROUTINE IS CALLED TO LOOKUP A LOGICAL NAME TABLE NAME.  THE FIRST MATCH
                        04AF  ; FOUND IS RETURNED.
                        04AF  ; ACCESS TO THE TABLE IS NOT CHECKED.
                        04AF  ;
                        04AF  ; INPUTS:
                        04AF  ;
                        04AF  ;       R1 = SEARCH ACCESS MODE IN LOW BYTE,
                        04AF  ;            CASELESS FLAG IN BIT 8,
                        04AF  ;            HIGH ORDER WORD 0.
                        04AF  ;       R2 = LENGTH OF TABLE NAME STRING.
                        04AF  ;       R3 = ADDRESS OF TABLE NAME STRING.
                        04AF  ;
                        04AF  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR AT LEAST READ
                        04AF  ;       ACCESS.
                        04AF  ;
                        04AF  ; OUTPUTS:
                        04AF  ;
                        04AF  ;       R0 LOW BIT CLEAR INDICATES SEARCH FAILURE.
                        04AF  ;
                        04AF  ;               R0 = SS$_NOLOGTAB - NO LOGICAL NAME TABLE NAME MATCH FOUND.
                        04AF  ;               R1 = JUNK.
                        04AF  ;
                        04AF  ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                        04AF  ;
                        04AF  ;               R1 = ADDRESS OF LOGICAL NAME TABLE HEADER.
                        04AF  ;
                        04AF  ;       REGISTERS R2 AND R3 ARE MODIFIED.
                        04AF  ;       REGISTERS R4 AND R5 ARE PRESERVED.
                        04AF  ;-
                        04AF
                        04AF  LNM$FIRSTTAB::                                  ;SEARCH FOR LOGICAL NAME TABLE
              55    DD  04AF          PUSHL   R5                              ;SAVE REGISTER
      5E      2C    C2  04B1          SUBL    #RT_K_LENGTH-4,SP               ;ALLOCATE RECURSIVE TABLE NAME CONTROL BLOCK
              51    DD  04B4          PUSHL   R1                              ;ACMODE, CASE FLAG
      55      5E    D0  04B6          MOVL    SP,R5                           ;ADDRESS OF BLOCK
            0134    30  04B9          BSBW    LNM$SETUP                       ;SETUP TABLE PROCESSING
      5E      30    C0  04BC          ADDL2   #RT_K_LENGTH,SP                 ;REMOVE TABLE SEARCH CONTROL BLOCK
              55  8ED0  04BF          POPL    R5                              ;RESTORE REGISTER
         05   50    E8  04C2          BLBS    R0,10$                          ;RETURN IF SUCCESSFUL
50     2294 8F      3C  04C5          MOVZWL  #SS$_NOLOGTAB,R0                ;ELSE SETUP TO RETURN APPROPRIATE ERROR
                    05  04CA  10$:    RSB                                     ;RETURN, STATUS IN R0
                        04CB
                        04CB  ;       .PAGE
```

```
04CB        .SBTTL  LNM$PRESEARCH   - FIND FIRST CANDIDATE NAME
04CB  ;+
04CB  ; LNM$PRESEARCH - FIND FIRST CANDIDATE LOGICAL NAME
04CB  ;
04CB  ; THIS ROUTINE IS CALLED TO SEARCH A LOGICAL NAME HASH TABLE FOR THE FIRST
04CB  ; CANDIDATE LOGICAL NAME MATCH.  IF A LOGICAL NAME TABLE IS SPECIFIED, THE
04CB  ; SEARCH CONTINUES TO A SPECIFIC NAME.
04CB  ;
04CB  ; INPUTS:
04CB  ;
04CB  ;       R1 = HASH TABLE ADDRESS (IF TABLE HEADER ADDRESS IS MISSING)
04CB  ;       R3 = ADDRESS OF NAME TRANSLATION (NT) BLOCK
04CB  ;
04CB  ;       NAME TABLE BLOCK REQUIREMENTS:
04CB  ;
04CB  ;               NT_W_RT         : MUST BE INITIALIZED
04CB  ;               NT_W_HASH       : MAY BE INITIALIZED OR 0
04CB  ;               NT_L_NAMLEN     : MUST BE INITIALIZED
04CB  ;               NT_L_NAMADR     : MUST BE INITIALIZED
04CB  ;               NT_L_TABID      : MAY BE INITIALIZED OR 0
04CB  ;               NT_L_THREAD     : UNINITIALIZED
04CB  ;
04CB  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR AT LEAST READ
04CB  ;       ACCESS.
04CB  ;
04CB  ; OUTPUTS:
04CB  ;
04CB  ;       R0 LOW BIT CLEAR INDICATES SEARCH FAILURE.
04CB  ;
04CB  ;               R0 = SS$_NOLOGNAM - NO LOGICAL NAME MATCH FOUND.
04CB  ;               R1 = ADDRESS OF LOGICAL NAME BLOCK ON WHICH SEARCH FAILED.
04CB  ;
04CB  ;               NAME TABLE BLOCK REQUIREMENTS:
04CB  ;
04CB  ;                       NT_W_RT         : UNCHANGED
04CB  ;                       NT_W_HASH       : VALID
04CB  ;                       NT_L_NAMLEN     : UNCHANGED
04CB  ;                       NT_L_NAMADR     : UNCHANGED
04CB  ;                       NT_L_TABID      : UNCHANGED
04CB  ;                       NT_L_THREAD     : ADDRESS OF LOGICAL NAME BLOCK
04CB  ;                                         PRECEEDING THE BLOCK ON WHICH THE
04CB  ;                                         SEARCH FAILED.
04CB  ;
04CB  ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
04CB  ;
04CB  ;               R1 = ADDRESS OF LOGICAL NAME BLOCK THAT CONTAINS MATCH.
04CB  ;
04CB  ;               NAME TABLE BLOCK REQUIREMENTS:
04CB  ;
04CB  ;                       NT_W_RT         : UNCHANGED
04CB  ;                       NT_W_HASH       : VALID
04CB  ;                       NT_L_NAMLEN     : UNCHANGED
04CB  ;                       NT_L_NAMADR     : UNCHANGED
04CB  ;                       NT_L_TABID      : UNCHANGED
04CB  ;                       NT_L_THREAD     : ADDRESS OF LOGICAL NAME BLOCK
04CB  ;                                         PRECEEDING THE BLOCK ON WHICH THE
04CB  ;                                         SEARCH SUCCEEDED.
```

```
                           04CB  ;
                           04CB  ;         R3,R4,R5 PRESERVED.
                           04CB  ;-
                           04CB            .ENABLE LSB
                           04CB
                           04CB  LNM$PRESEARCH::                              ;SEARCH FOR LOGICAL NAME
       50    0C A3    D0   04CB            MOVL    NT_L_TABID(R3),R0           ;ADDRESS OF TABLE HEADER
                04    13   04CF            BEQL    10$                         ;BRANCH IF NOT SPECIFIED
       51    01 A0    D0   04D1            MOVL    LNMTH$L_HASH(R0),R1         ;HASH TABLE ADDRESS
       50    02 A3    3C   04D5  10$:      MOVZWL  NT_W_HASH(R3),R0            ;HASH FUNCTION AVAILABLE?
                10    12   04D9            BNEQ    20$                         ;BRANCH IF YES
                51    DD   04DB            PUSHL   R1                          ;SAVE HASH TABLE ADDRESS
       50    04 A3    7D   04DD            MOVQ    NT_L_NAMLEN(R3),R0          ;NAME DESCRIPTOR
              008A    30   04E1            BSBW    LNM$HASH                    ;COMPUTE HASH FUNCTION
    02 A3       50    B0   04E4            MOVW    R0,NT_W_HASH(R3)            ;SAVE HASH FUNCTION
                51  8ED0   04E8            POPL    R1                          ;RESTORE HASH TABLE ADDRESS
         50     61    CA   04EB  20$:      BICL2   LNMHSH$L_MASK(R1),R0        ;MASK OFF UNWANTED BITS OF HASH FUNCTION
  51     0C   A140    DE   04EE            MOVAL   LNMHSH$K_BUCKET(R1)[R0],-   ;COMPUTE ADDRESS OF HASH BUCKET AND
                           04F3                    R1                          ;STORE ITS ADDRESS AS ADDRESS OF
                           04F3            ASSUME  LNMB$L_FLINK,EQ,0           ;PREVIOUS LNMB IN NT_L_THREAD
                0D    11   04F3            BRB     30$                         ;RUN DOWN HASH CHAIN AND RETURN
                           04F5
                           04F5  ;         .PAGE
```

```
                      04F5           .SBTTL  LNM$CONTSEARCH  - FIND NEXT CANDIDATE NAME
                      04F5    ;+
                      04F5    ; LNM$CONTSEARCH - FIND NEXT CANDIDATE LOGICAL NAME
                      04F5    ;
                      04F5    ; THIS ROUTINE IS CALLED TO SEARCH A LOGICAL NAME HASH BUCKET FOR THE NEXT
                      04F5    ; CANDIDATE LOGICAL NAME MATCH.
                      04F5    ;
                      04F5    ; INPUTS:
                      04F5    ;
                      04F5    ;       R3 = ADDRESS NAME TABLE BLOCK.
                      04F5    ;
                      04F5    ;       NAME TABLE BLOCK REQUIREMENTS:
                      04F5    ;
                      04F5    ;               NT_W_R5         : MUST BE INITIALIZED
                      04F5    ;               NT_W_HASH       : MUST BE INITIALIZED
                      04F5    ;               NT_L_NAMLEN     : MUST BE INITIALIZED
                      04F5    ;               NT_L_NAMADR     : MUST BE INITIALIZED
                      04F5    ;               NT_L_TABID      : MUST BE INITIALIZED
                      04F5    ;               NT_L_THREAD     : MUST BE INITIALIZED
                      04F5    ;                       ADDRESS OF PREVIOUS LNMB$ BLOCK -
                      04F5    ;                       SEARCH CONTINUES WITH THE FOLLOWING ENTRY.
                      04F5    ;
                      04F5    ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR AT LEAST READ
                      04F5    ;       ACCESS.
                      04F5    ;
                      04F5    ; OUTPUTS:
                      04F5    ;
                      04F5    ;       R0 LOW BIT CLEAR INDICATES SEARCH FAILURE.
                      04F5    ;
                      04F5    ;               R0 = SS$_NOLOGNAM - NO LOGICAL NAME MATCH FOUND.
                      04F5    ;               R1 = ADDRESS OF LOGICAL NAME BLOCK ON WHICH SEARCH FAILED.
                      04F5    ;
                      04F5    ;               NT_L_THREAD CONTAINS ADDRESS OF PREVIOUS LNMB BLOCK
                      04F5    ;
                      04F5    ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                      04F5    ;
                      04F5    ;               R1 = ADDRESS OF LOGICAL NAME BLOCK THAT CONTAINS MATCH.
                      04F5    ;
                      04F5    ;               NT_L_THREAD CONTAINS ADDRESS OF PREVIOUS LNMB BLOCK
                      04F5    ;
                      04F5    ;       R3,R4,R5 ARE PRESERVED.
                      04F5    ;-
                      04F5
                      04F5    LNM$CONTSEARCH::                                        ;SEARCH FOR LOGICAL NAME
                      04F5            ASSUME  LNMB$L_FLINK,EQ,0
   51    10 A3    D0  04F5            MOVL    NT_L_THREAD(R3),R1                    ;ADDRESS OF PREVIOUS LOGICAL NAME BLOCK
   09 63    0A    E0  04F9            BBS     #NT_V_MODIFY,(R3),35$                 ;CAN HASH BUCKET CHANGE?
      51    61    D0  04FD            MOVL    LNMB$L_FLINK(R1),R1                   ;IF NOT THEN RETRIEVE ADDRESS  NEXT LNMB
      54    11        0500            BRB     55$                                   ;AND SKIP FIRST CMPC3
                      0502
   10 A3    51    D0  0502    30$:    MOVL    R1,NT_L_THREAD(R3)                    ;SAVE ADDRESS OF PREVIOUS BLOCK
      51    61    D0  0506    35$:    MOVL    LNMB$L_FLINK(R1),R1                   ;GET ADDRESS OF NEXT LOGICAL NAME BLOCK
      45    13        0509            BEQL    70$                                   ;BRANCH IF NO NEXT BLOCK
11 A1    04 A3    91  050B            CMPB    NT_L_NAMLEN(R3), -                    ;LENGTH'S MATCH?
                      0510                    LNMB$T_NAME(R1)
      F0    1A        0510            BGTRU   30$                                   ;KEEP LOOKING DOWN CHAIN
      3C    1F        0512            BLSSU   70$                                   ;NO MATCH
```

```
                                                            G  1
LNMSUB                          - LOGICAL NAME RELATED SUBROUTINES      16-SEP-1984 00:30:35  VAX/VMS Macro V04-00        Page  47
V04-000                         LNM$CONTSEARCH - FIND NEXT CANDIDATE NAM  5-SEP-1984 03:44:03  [SYS.SRC]LNMSUB.MAR;1            (22)

                     53   DD  0514  2162         PUSHL   R3                          ;SAVE SEARCH PARAMETERS
                     51   DD  0516  2163         PUSHL   R1
   12 A1    08 B3   04 A3   29  0518  2164         CMPC3   NT_L_NAMLEN(R3), -          ;LOGICAL NAME STRINGS MATCH?
                              051F  2165                 @NT_L_NAMADR(R3), -
                              051F  2166                 LNMB$T_NAME+1(R1)
         22 04 BE    08   E1  051F  2167         BBC     #NT_V_CASE,@4(SP),50$       ;BRANCH IF CASED SEARCH
                     20   13  0524  2168 40$:    BEQL    50$                         ;BRANCH ON MATCH
              50     81   9A  0526  2169         MOVZBL  (R1)+,R0                    ;GET CASELESS CHARACTER
      7E   00000000'GF40  90  0529  2170         MOVB    G^EXE$UPCASE_DAT[R0],-(SP)
              50     83   9A  0531  2171         MOVZBL  (R3)+,R0                    ;GET CASELESS CHARACTER
       00000000'GF40 8E   91  0534  2172         CMPB    (SP)+,G^EXE$UPCASE_DAT[R0]
                     08   12  053C  2173         BNEQ    50$                         ;BRANCH IF DIFFERENT IN MORE THAN CASE
                     52   D7  053E  2174         DECL    R2                          ;PASS OVER THESE CHARACTERS
           63     61 52   29  0540  2175         CMPC3   R2,(R1),(R3)                ;RESUME COMPARISON
                     DE   11  0544  2176         BRB     40$                         ;CONTINUE CASELESS COMPARISON
                     0A   BA  0546  2177 50$:    POPR    #^M<R1,R3>                  ;RETRIEVE SEARCH PARAMETERS
                     0C   13  0548  2178         BEQLU   55$                         ;IF EQL FOUND IT
                     B6   1A  054A  2179         BGTRU   30$                         ;IF GTR KEEP SEARCHING
              B2 63  08   E0  054C  2180         BBS     #NT_V_CASE,(R3),30$         ;BRANCH TO CONTINUE IF CASED SEARCH
           50   01BC 8F   3C  0550  2181 70$:    MOVZWL  #SS$_NOLOGNAM,R0            ;SET NO LOGICAL NAME MATCH
                          05  0555  2182         RSB                                 ;EXIT
                              0556  2183
              50   0C A3  D0  0556  2184 55$:    MOVL    NT_L_TABID(R3),R0           ;RETRIEVE TARGET TABLE HEADER ADDRESS
                     0E   13  055A  2185         BEQL    60$                         ;DONE IF IGNORING TABLE
              0C A1  50   D1  055C  2186         CMPL    R0,LNMB$L_TABLE(R1)         ;COMPARE TABLE
                     A0   1A  0560  2187         BGTRU   30$                         ;CONT IF NOT YET REACHED TARGET TABLE
                     EC   1F  0562  2188         BLSSU   70$                         ;DONE IF PASSED TARGET TABLE
              0B A1  63   91  0564  2189         CMPB    NT_B_ACMODE(R3), -          ;COMPARE ACCESS MODES
                              0568  2190                 LNMB$B_ACMODE(R1)
                     98   1F  0568  2191         BLSSU   30$                         ;CONT IF NOT YET REACHED ACCESS MODE
                              056A  2192
                 50  01   D0  056A  2193 60$:    MOVL    #SS$_NORMAL,R0              ;MATCH FOUND
                          05  056D  2194         RSB                                 ;EXIT
                              056E  2195
                              056E  2196         .DISABLE        LSB
                              056E  2197 ;       .PAGE
```

```
                                056E          .SBTTL  LNM$HASH        - HASHING ALGORITHM
                                056E  ;+
                                056E  ; LNM$HASH - THE HASH FUNCTION
                                056E  ;
                                056E  ; INPUTS:
                                056E  ;         R0 = LENGTH OF NAME.
                                056E  ;         R1 = ADDRESS OF NAME.
                                056E  ;
                                056E  ; OUTPUTS:
                                056E  ;         R0 = RETURN HASH FUNCTION.
                                056E  ;         R1, R2 SCRATCHED.
                                056E  ;         R3,R4,R5 PRESERVED.
                                056E  ;-
                                056E
                                056E  LNM$HASH::
                      53   DD   056E          PUSHL   R3                        ;SAVE REGISTERS
                      50   DD   0570          PUSHL   R0
         53   50   FE 8F   78   0572          ASHL    #-2,R0,R3                 ;DIVIDE BY FOUR
                      12   13   0577          BEQL    20$                       ;NOT EVEN THAT LONG
           20202020 8F     CB   0579  10$:    BICL3   #^A/AAAA/\^A/aaaa/,-      ;FETCH CASELESS CHARACTERS
                 52   81        057F                  (R1)+,R2
                 50   52   CC   0581          XORL    R2,R0                     ;XOR IN THE CHARACTERS
            50   50   09   9C   0584          ROTL    #9,R0,R0                  ;SCRAMBLE A BIT
                   EE 53   F5   0588          SOBGTR  R3,10$                    ;LOOP UNTIL DONE
   53   8E   FFFFFFFC 8F   CB   058B  20$:    BICL3   #-4,(SP)+,R3              ;SAVE REMAINDER OF DIVIDE BY FOUR
                      0B   11   0593          BRB     40$
            52   81   20   8B   0595  30$:    BICB3   #^A/A/\^A/a/,(R1)+,R2     ;FETCH A CASELESS CHARACTER
                 50   52   8C   0599          XORB    R2,R0                     ;XOR IN A CHARACTER
            50   50   0D   9C   059C          ROTL    #13,R0,R0                 ;SCRAMBLE
                   F2 53   F4   05A0  40$:    SOBGEQ  R3,30$                    ;LOOP UNTIL DONE
         50   71279461 8F  C4   05A3          MULL    #^X71279461,R0            ;MULT BY A FUNNY NUMBER
            50   50   13   9C   05AA          ROTL    #32-13,R0,R0              ;GET SIGNIFICANT BITS
                   53 8ED0      05AE          POPL    R3                        ;RESTORE REGISTERS
                        05      05B1          RSB
                                05B2  ;       .PAGE
```

```
                          05B2        .SBTTL  LNM$LOOKUP      - LOOKUP TABLE NAME
                          05B2  ;+
                          05B2  ; LNM$LOOKUP - LOOKUP TABLE NAME
                          05B2  ;
                          05B2  ; THIS ROUTINE IS CALLED TO LOOKUP A LOGICAL NAME TABLE NAME.
                          05B2  ;
                          05B2  ; INPUTS:
                          05B2  ;
                          05B2  ;       R0 = HASH CODE VALUE OF LOGICAL NAME TABLE STRING (OR 0 IF NOT KNOWN)
                          05B2  ;       R2 = LENGTH OF LOGICAL NAME TABLE STRING.
                          05B2  ;       R3 = ADDRESS OF LOGICAL NAME TABLE STRING.
                          05B2  ;       R5 = ADDRESS OF RECURSION TABLE SEARCH CONTROL BLOCK
                          05B2  ;
                          05B2  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR AT LEAST READ
                          05B2  ;       ACCESS.
                          05B2  ;
                          05B2  ; OUTPUTS:
                          05B2  ;
                          05B2  ;       R1 POINTS TO THE NAME BLOCK.
                          05B2  ;
                          05B2  ; R2 AND R3 ARE MODIFIED.
                          05B2  ;-
                          05B2
                          05B2  LNM$LOOKUP:
                55   DD   05B2          PUSHL   R5                      ;SAVE REGISTER
                7E   7C   05B4          CLRQ    -(SP)                   ;NO NAME BLOCK ADDRESS
                          05B6                                          ;SPACE FOR DIRECTORY TABLE ADDRESS
          7E    52   7D   05B6          MOVQ    R2,-(SP)                ;ADDRESS AND LENGTH OF NAME
          7E    50   B0   05B9          MOVW    R0,-(SP)                ;HASH CODE VALUE OF TABLE NAME STRING
          7E    65   B0   05BC          MOVW    RT_W_R5(R5),-(SP)       ;CASE FLAG AND AC.CESS MODE
          53    5E   D0   05BF          MOVL    SP,R3                   ;ADDRESS OF BLOCK
55  00000000'EF      DE   05C2          MOVAL   L^LNM$AL_DIRTBL,R5      ;ADDR OF SYSTEM DIRECTORY TABLE ADDRESS
      11 5E     1F   E0   05C9          BBS     #31,SP,10$              ;BRANCH IF SMALL PROCESS
     50    04 B5     D0   05CD          MOVL    @4(R5),R0               ;ADDRESS OF PROCESS DIRECTORY TABLE
                0B   13   05D1          BEQL    10$                     ;SKIP IF NO TABLE DEFINED
             0C A0   D0   05D3          MOVL    LNMB$L_TABLE(R0),-      ;ADDRESS OF TABLE HEADER OF PROCESS
             0C A3        05D6                  NT_L_TABID(R3)          ;DIRECTORY TABLE
              FEF0   30   05D8          BSBW    LNM$PRESEARCH           ;SEARCH PROCESS DIRECTORY
             0B 50   E8   05DB          BLBS    R0,20$                  ;BRANCH IF NAME FOUND IN PROCESS SPACE
          50    95   D0   05DE  10$:    MOVL    @(R5)+,R0               ;ADDRESS OF SYSTEM DIRECTORY TABLE
             0C A0   D0   05E1          MOVL    LNMB$L_TABLE(R0),-      ;ADDRESS OF TABLE HEADER OF SYSTEM
             0C A3        05E4                  NT_L_TABID(R3)          ;DIRECTORY TABLE
              FEE2   30   05E6          BSBW    LNM$PRESEARCH           ;SEARCH SYSTEM DIRECTORY AND RETURN
          5E    14   C0   05E9  20$:    ADDL    #NT_K_LENGTH,SP         ;FLUSH TRANSLATION BLOCK
             55 8ED0      05EC          POPL    R5                      ;RESTORE REGISTER
                     05   05EF  LRSB:   RSB
                          05F0
                          05F0  ;       .PAGE
```

```
                      05F0          .SBTTL  LNM$SETUP       - SETUP TO PROCESS TABLE NAME
                      05F0  ;+
                      05F0  ; LNM$SETUP - SETUP TO PROCESS LOGICAL NAME TABLE NAME
                      05F0  ;
                      05F0  ; THIS ROUTINE IS CALLED TO SETUP TO PROCESS A LOGICAL NAME TABLE NAME.
                      05F0  ; TABLE SEARCHING IS INITIALIZED.
                      05F0  ;
                      05F0  ; INPUTS:
                      05F0  ;
                      05F0  ;       R2 = LENGTH OF LOGICAL NAME TABLE STRING.
                      05F0  ;       R3 = ADDRESS OF LOGICAL NAME TABLE STRING.
                      05F0  ;       R5 = ADDRESS OF RECURSIVE TABLE NAME TRANSLATION BLOCK
                      05F0  ;               WITH RT_W_R5 FIELDS INITIALIZED
                      05F0  ;
                      05F0  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR AT LEAST READ
                      05F0  ;       ACCESS.
                      05F0  ;
                      05F0  ; OUTPUTS:
                      05F0  ;
                      05F0  ;       R0 CONTAINS ERROR STATUS FROM SEARCHING.
                      05F0  ;               R0 = SS$_NOLOGNAM - NO LOGICAL NAME MATCH FOUND.
                      05F0  ;               R2,R3,R5 ARE MODIFIED.
                      05F0  ;               R4 IS PRESERVED.
                      05F0  ;
                      05F0  ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                      05F0  ;               R1 = ADDRESS OF LOGICAL NAME TABLE HEADER.
                      05F0  ;               R2 = LENGTH OF ACTUAL LOGICAL NAME TABLE STRING
                      05F0  ;               R3 = ADDRESS OF ACTUAL LOGICAL NAME TABLE STRING
                      05F0  ;               R5 = ADDRESS OF CONTROL BLOCK.
                      05F0  ;-
                      05F0
                      05F0  LNM$SETUP::
              02  8A  05F0          BICB2   #RT_M_TERM,-            ;CLEAR LAST TRANSLATION BIT
           01 A5      05F2                  RT_B_FLAGS(R5)
           02 A5  94  05F4          CLRB    RT_B_DEPTH(R5)          ;INITIALIZE RECURSION DEPTH TO 0
           FF 8F  90  05F7          MOVB    #RT_C_MAXTRIES,-        ;INITIALIZE MAXIMUM NUMBER OF TRIES
           03 A5      05FA                  RT_B_TRIES(R5)
              50  D4  05FC          CLRL    R0                      ;HASH CODE VALUE OF INITIAL TABLE NAME
                      05FE                                          ;STRING IS NOT KNOWN
            FFB1  30  05FE          BSBW    LNM$LOOKUP              ;LOOKUP THE INITIAL NAME
           EB 50  E9  0601          BLBC    R0,LRSB                 ;NO SUCH NAME
   08 A5   51 11  C1  0604          ADDL3   #LNMB$T_NAME,R1,-       ;SAVE INITIAL LNMB IN RECURSION TABLE
                      0609                  RT_L_STACK(R5)          ; AS THE STARTING POINT OF TRANSLATIONS
            00E4  30  0609          BSBW    LNM$TBL_CACHE           ;CHECK THE TABLE TRANSLATION CACHE
     04 A5    50  D0  060C          MOVL    R0,RT_L_CACHEPTR(R5)    ;SAVE CACHE POINTER
              57  13  0610          BEQL    LNM$TABLE_SRCH          ;USE LONG WAY IF NO CACHE ENTRY
     0B A0    01  8E  0612          MNEGB   #1,LNMC$B_CACHEINDX(R0) ;START WITH INITIAL ENTRY
                      0616  ;       BRB     LNM$TABLE               ; AND DROP INTO LNM$TABLE
                      0616  ;
                      0616  ;       .PAGE
                      0616          .SBTTL  LNM$TABLE       - PROCESS LOGICAL NAME TABLE
                      0616   ;+
                      0616   ; LNM$TABLE - PROCESS LOGICAL NAME TABLE NAME
                      0616   ;
                      0616   ; THIS ROUTINE IS CALLED TO PROCESS A LOGICAL NAME TABLE NAME.
                      0616   ; THE TABLE NAME TRANSLATION CACHE IS USED IF POSSIBLE, ELSE
                      0616   ; THE NAME IS RECURSIVELY TRANSLATED.  A CALLBACK IS PERFORMED
                      0616   ; FOR EVERY TABLE THAT IS FOUND.
                      0616   ;
                      0616   ; A BASIC ASSUMPTION THAT THIS ROUTINE MAKES IS THAT IT IS CALLED FIRST
                      0616   ; THROUGH LNM$SETUP TO INITIALIZE THE RECURSION TABLE BEFORE BEING CALLED
                      0616   ; DIRECTLY SUBSEQUENT TIMES.
                      0616   ;
                      0616   ; INPUTS:
                      0616   ;
                      0616   ;       R5 = ADDRESS OF TABLE NAME TRANSLATION BLOCK
                      0616   ;               ALL FIELDS OF THE BLOCK MUST BE INITIALIZED.
                      0616   ;
                      0616   ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR AT LEAST READ
                      0616   ;       ACCESS.
                      0616   ;
                      0616   ; OUTPUTS:
                      0616   ;
                      0616   ;       R0 CONTAINS ERROR STATUS FROM SEARCHING.
                      0616   ;               R0 = SS$_NOLOGNAM - NO LOGICAL NAME MATCH FOUND.
                      0616   ;               R2,R3,R5 ARE MODIFIED.
                      0616   ;               R4 IS PRESERVED.
                      0616   ;
                      0616   ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                      0616   ;               R1 = ADDRESS OF LOGICAL NAME TABLE HEADER.
                      0616   ;               R2 = LENGTH OF ACTUAL LOGICAL NAME TABLE STRING
                      0616   ;               R3 = ADDRESS OF ACTUAL LOGICAL NAME TABLE STRING
                      0616   ;               R5 = ADDRESS OF CONTROL BLOCK.
                      0616   ;
                      0616   ;-
                      0616
                      0616  LNM$TABLE::
52    04 A5     D0    0616          MOVL    RT L_CACHEPTR(R5),R2    ;GET CACHE POINTER
         4D     13    061A          BEQL    LNM$TABLE_SRCH          ;USE RECURSIVE METHOD IF NO CACHE
      0B A2     96    061C          INCB    LNMC$B_CACHEINDX(R2)    ;GO TO NEXT ENTRY
53    0B A2     9A    061F          MOVZBL  LNMC$B_CACHEINDX(R2),R3 ;EXTRACT INDEX NUMBER
   1A    53     91    0623          CMPB    R3,#LNMC$K_NUM_ENTRIES  ;OFF THE END?
         3E     1E    0626          BGEQU   40$                     ;NOPE, SO USE THIS ENTRY
51    18 A243   D0    0628          MOVL    LNMC$L_ENTRY(R2)[R3],R1 ;GET ENTRY
         10     13    062D          BEQL    20$                     ;IS THERE ONE?
                      062F
50    51    01  C1    062F          ADDL3   #1,R1,R0                ;WAS THIS THE END FLAG?
            04  13    0633          BEQL    10$                     ;THEN WE GOT ONE
      50    01  3C    0635          MOVZWL  #SS$_NORMAL,R0          ;SUCCESS!
                05    0638          RSB
                      0639
50    01BC 8F   3C    0639  10$:    MOVZWL  #SS$_NOLOGNAM,R0        ;RETURN TABLES ALL DONE
                05    063E          RSB
                      063F
                      063F   ;
                      063F   ; NO ENTRY - TWO POSSIBLE CASES:
                      063F   ;  1.   WE JUST RAN OFF THE END OF THE VALID ONES AND
```

```
                          063F  ;        NEED TO GO BACK TO THE BEGINNING AND REBUILD IT ALL.
                          063F  ;        IN THIS CASE THE RECURSION TABLE IS STILL IN THE INITIALIZED STATE
                          063F  ;        AND THE RECURSION DEPTH IS ZERO.
                          063F  ;
                          063F  ;   2.   WE ARE BUILDING THEM AS WE GO.
                          063F  ;        IN THIS CASE THE RECURSION TABLE IS CURRENT AND THE CACHE INDEX
                          063F  ;        IS CORRECT.
                          063F  ;
                          063F
           02 A5    95    063F  20$:     TSTB    RT_B_DEPTH(R5)          ;RECURSION DEPTH 0?
              0E    12    0642           BNEQ    30$                     ;NOPE
           0C A2    C1    0644           ADDL3   LNMC$L_TBLADDR(R2),-    ;GET POINTER TO TABLE NAME
       51     11          0647                   #LNMB$T_NAME,R1
    08 A5     51    D1    0649           CMPL    R1,RT_L_STACK(R5)       ;INITIAL STATE?
              03    12    064D           BNEQ    30$                     ;NO, SO PROCEED
           0B A2    94    064F           CLRB    LNMC$B_CACHEINDX(R2)    ;GO BACK TO START
              15    10    0652  30$:     BSBB    LNM$TABLE_SRCH          ;FIND NEXT (OR FIRST)
    52     04 A5    D0    0654           MOVL    RT_L_CACHEPTR(R5),R2    ;GET CACHE POINTER
    53     0B A2    9A    0658           MOVZBL  LNMC$B_CACHEINDX(R2),R3 ;EXTRACT INDEX NUMBER
  18 A243     51    D0    065C           MOVL    R1,LNMC$L_ENTRY(R2)[R3] ;STORE TABLE HEADER ADDR
      1C A243       D4    0661           CLRL    LNMC$L_ENTRY+4(R2)[R3]  ;CLEAR NEXT
                    05    0665           RSB                             ;RETURN
                          0666
           04 A5    D4    0666  40$:     CLRL    RT_L_CACHEPTR(R5)       ;GIVE UP ON THE CACHE
                          0669  ;        BRB     LNM$TABLE_SRCH          ;AND USE THE LONG METHOD
                          0669
                          0669  ;        .PAGE
```

```
                        0669          .SBTTL  LNM$TABLE_SRCH  - PROCESS LOGICAL NAME TABLE
                        0669    ;+
                        0669    ; LNM$TABLE_SRCH - PROCESS LOGICAL NAME TABLE NAME
                        0669    ;
                        0669    ; THIS ROUTINE IS CALLED TO PROCESS A LOGICAL NAME TABLE NAME.
                        0669    ; THE NAME IS RECURSIVELY TRANSLATED.  A CALLBACK IS PERFORMED
                        0669    ; FOR EVERY TABLE THAT IS FOUND.
                        0669    ;
                        0669    ; A BASIC ASSUMPTION THAT THIS ROUTINE MAKES IS THAT IT IS CALLED FIRST
                        0669    ; THROUGH LNM$SETUP TO INITIALIZE THE RECURSION TABLE BEFORE BEING CALLED
                        0669    ; DIRECTLY SUBSEQUENT TIMES.
                        0669    ;
                        0669    ; INPUTS:
                        0669    ;
                        0669    ;       R5 = ADDRESS OF TABLE NAME TRANSLATION BLOCK
                        0669    ;               ALL FIELDS OF THE BLOCK MUST BE INITIALIZED.
                        0669    ;
                        0669    ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR AT LEAST READ
                        0669    ;       ACCESS.
                        0669    ;
                        0669    ; OUTPUTS:
                        0669    ;
                        0669    ;       R0 CONTAINS ERROR STATUS FROM SEARCHING.
                        0669    ;               R0 = SS$_NOLOGNAM - NO LOGICAL NAME MATCH FOUND.
                        0669    ;               R2,R3,R5 ARE MODIFIED.
                        0669    ;               R4 IS PRESERVED.
                        0669    ;
                        0669    ;       R0 LOW BIT SET INDICATES SUCCESS WITH:
                        0669    ;               R1 = ADDRESS OF LOGICAL NAME TABLE HEADER.
                        0669    ;               R2 = LENGTH OF ACTUAL LOGICAL NAME TABLE STRING
                        0669    ;               R3 = ADDRESS OF ACTUAL LOGICAL NAME TABLE STRING
                        0669    ;               R5 = ADDRESS OF CONTROL BLOCK.
                        0669    ;
                        0669    ;-
                        0669
                        0669    LNM$TABLE_SRCH:
   01 A5    02    8A    0669    10$:    BICB2   #RT_M_TERM,RT_B_FLAGS(R5) ;CLEAR TERMINAL SEEN BIT
 50    02 A5    98    066D            CVTBL   RT_B_DEPTH(R5),R0         ;RECURSION DEPTH
          65    19    0671            BLSS    60$                       ;BRANCH IF NOTHING LEFT TO SCAN
51   08 A540    D0    0673            MOVL    RT_L_STACK(R5)[R0],R1     ;ADDRESS OF PREVIOUS TRANSLATION STRING
                        0678
       03 A5    97    0678    20$:    DECB    RT_B_TRIES(R5)            ;DECREMENT NUMBER OF TRIES LEFT
          64    13    067B            BEQL    70$                       ;DONE IF TRIES REACHES ZERO
                        067D
      50  81    9A    067D            MOVZBL  (R1)+,R0                  ;LENGTH OF PREVIOUS TRANSLATION STRING
      51  50    C0    0680            ADDL2   R0,R1                     ;POSITION PAST PREVIOUS TRANSLATION
                        0683
          02    E0    0683            BBS     #LNMX$V_XEND,-            ;LAST TRANSLATION?
       4C 61          0685                    LNMX$B_FLAGS(R1),50$      ;YES - GO DECREMENT DEPTH AND CONTINUE
 52    01 A1    98    0687            CVTBL   LNMX$B_INDEX(R1),R2       ;GET TRANSLATION INDEX
          34    19    068B            BLSS    40$                       ;BRANCH IF SPECIAL VALUE
   57 65    09    E0    068D            BBS     #RT_V_TERM,(R5),80$       ;ERROR IF EXPECTING TABLE HEADER
                        0691
          0A    91    0691            CMPB    #LNM$C_MAXDEPTH,-         ;RECURSED TOO DEEPLY?
       02 A5          0693                    RT_B_DEPTH(R5)            ;ERROR OF OVERLY DEEP
          4A    15    0695            BLEQ    70$
                        0697
```

```
           50    02 A5    98  0697  2477         CVTBL   RT_B_DEPTH(R5),R0         ;RECURSION DEPTH
           53    04 A1    9E  069B  2478         MOVAB   LNMX$T_XLATION(R1),R3     ;TRANSLATION COUNTED STRING ADDRESS
         08 A540    53    D0  069F  2479         MOVL    R3,RT_C_STACK(R5)[R0]     ;SAVE RECURSION INFO BEFORE LOOKUP
                 02 A5    96  06A4  2480         INCB    RT_B_DEPTH(R5)            ;INCREMENT RECURSION DEPTH FOR NEXT TIME
              52    83    9A  06A7  2481         MOVZBL  (R3)+,R2                  ;TRANSLATION STRING LENGTH AND ADDRESS
           50    02 A1    3C  06AA  2482         MOVZWL  LNMX$W_HASH(R1),R0        ;RETRIEVE TRANSLATION'S HASH CODE VALUE
                              06AE  2483
                    01    E1  06AE  2484         BBC     #LNMX$V_TERMINAL,-        ;IS THIS TRANSLATION MARKED TERMINAL?
                 04 61        06B0  2485                 LNMX$B_FLAGS(R1),30$      ;NO  - GO DO LOOKUP
                    02    88  06B2  2486         BISB2   #RT_M_TERM,-              ;YES - ALLOW ONE AND ONLY ONE MORE
                 01 A5        06B4  2487                 RT_B_FLAGS(R5)            ;       TRANSLATION
                              06B6  2488
                  FEF9    30  06B6  2489 30$:    BSBW    LNM$LOOKUP                ;LOOKUP LOGICAL NAME TABLE NAME
                 17 50    E9  06B9  2490         BLBC    R0,50$                    ;GO DECR DEPTH AND CONTINUE IF NOT FOUND
              51    11    C0  06BC  2491         ADDL2   #LNMB$T_NAME,R1           ;POINT TO COUNTED NAME STRING
                    B7    11  06BF  2492         BRB     20$                       ;CONTINUE WITH FIRST TRANSLATION
                              06C1  2493
              51    04    C0  06C1  2494 40$:    ADDL2   #LNMX$T_XLATION,R1        ;ADDRESS OF COUNTED TRANSLATION STRING
           82 8F    52    91  06C4  2495         CMPB    R2,#LNMX$C_TABLE          ;TABLE NAME?
                    AE    12  06C8  2496         BNEQ    20$                       ;IGNORE THIS TRANSLATION
                    51    D6  06CA  2497         INCL    R1                        ;POSITION TO TABLE HEADER
                 02 A5    97  06CC  2498         DECB    RT_B_DEPTH(R5)            ;DECREMENT RECURSION DEPTH
              50    01    D0  06CF  2499         MOVL    #SS$_NORMAL,R0            ;NORMAL STATUS
                          05  06D2  2500         RSB                               ;RETURN WITH R1 = TABLE HEADER ADDRESS
                              06D3  2501
                 02 A5    97  06D3  2502 50$:    DECB    RT_B_DEPTH(R5)            ;DECREMENT RECURSION DEPTH
                    91    18  06D6  2503         BGEQ    10$                       ;RESUME
                              06D8  2504
              51    01    CE  06D8  2505 60$:    MNEGL   #1,R1                     ;FLAG LAST TABLE
        50    01BC  8F    3C  06DB  2506         MOVZWL  #SS$_NOLOGNAM,R0          ;HAVE SCANNED ALL TABLE
                          05  06E0  2507         RSB
        50    0374  8F    3C  06E1  2508 70$:    MOVZWL  #SS$_TOOMANYLNAM,R0       ;RECURSION TOO DEEP
                    05    11  06E6  2509         BRB     90$                       ;RETURN
        50    015C  8F    3C  06E8  2510 80$:    MOVZWL  #SS$_IVLOGTAB,R0          ;INVALID TABLE NAME
                    51    D4  06ED  2511 90$:    CLRL    R1
                          05  06EF  2512         RSB
                              06F0  2513
                              06F0  2514 ;       .PAGE
```

```
                         06F0         .SBTTL  LNM$TBL_CACHE   - SEARCH LOGICAL NAME TABLE TRANSLATION CACHE
                         06F0  ;+
                         06F0  ; LNM$TBL_CACHE - SEARCH LOGICAL NAME TABLE TRANSLATION CACHE
                         06F0  ;
                         06F0  ; THIS ROUTINE IS CALLED TO SEARCH THE LOGICAL NAME TABLE NAME
                         06F0  ; TRANSLATION CACHE.  IF A CACHE ENTRY EXISTS FOR THIS LOGICAL NAME,
                         06F0  ; THEN THE ENTRY IS VALIDATED AND RETURNED.   IF INVALID OR NO ENTRY,
                         06F0  ; A NEW ENTRY IS SELECTED, INITIALIZED AND RETURNED.
                         06F0  ;
                         06F0  ; INPUTS:
                         06F0  ;
                         06F0  ;       R1 = ADDRESS OF TABLE NAME LOGICAL NAME BLOCK
                         06F0  ;
                         06F0  ;       IT IS ASSUMED THAT THE LOGICAL NAME MUTEX IS LOCKED FOR AT LEAST READ
                         06F0  ;       ACCESS.
                         06F0  ;
                         06F0  ; OUTPUTS:
                         06F0  ;
                         06F0  ;       R0 CONTAINS ADDRESS OF CACHE ENTRY TO USE OR 0
                         06F0  ;          CACHE ENTRY MAY OR MAY NOT BE VALID
                         06F0  ;
                         06F0  ;-
                         06F0
                         06F0  LNM$TBL_CACHE:
        50 5E   1F   E0  06F0         BBS     #31,SP,100$               ;NO CACHE IF NO P1 SPACE
                         06F4
   52   00000000'9F  9E  06F4         MOVAB   @#CTL$GQ_LNMTBLCACHE,R2 ;GET QUEUE HEADER ADDR
             50  52  D0  06FB         MOVL    R2,R0                     ;POINT TO FIRST ENTRY
             50  60  D0  06FE  10$:   MOVL    LNMC$L_FLINK(R0),R0       ;GET NEXT ENTRY
             50  52  D1  0701         CMPL    R2,R0                     ;BACK TO QUEUE HEADER?
                 38  13  0704         BEQL    80$                       ;YES, THEN MAKE A NEW ONE
          0C A0  51  D1  0706         CMPL    R1,LNMC$L_TBLADDR(R0)     ;IS THIS THE ONE?
                 F2  12  070A         BNEQ    10$                       ;TRY NEXT
                         070C
             50  60  0F  070C         REMQUE  (R0),R0                   ;REMOVE ENTRY
              10 A0  D1  070F         CMPL    LNMC$L_PROCDIRSEQ(R0),- ;IS PROCESS DIRECTORY VALID?
        00000000'9F      0712                 @#CTL$GL_LNMDIRSEQ
                 0A  12  0717         BNEQ    40$                       ;NOPE, MUST RE-INIT ENTRY
              14 A0  D1  0719         CMPL    LNMC$L_SYSDIRSEQ(R0),-  ;IS SYSTEM DIRECTORY VALID?
        00000000'9F      071C                 @#LNM$GL_SYSDIRSEQ
                 17  13  0721         BEQL    50$                       ;A CACHE HIT!
          0C A0  51  D0  0723  40$:   MOVL    R1,LNMC$L_TBLADDR(R0)     ;SET TABLE NAME ADDR
        00000000'9F  D0  0727         MOVL    @#CTL$GL_LNMDIRSEQ,-      ;SET NEW PROCESS DIRECTORY SEQ NUM.
              10 A0      072D                 LNMC$L_PROCDIRSEQ(R0)
        00000000'9F  D0  072F         MOVL    @#LNM$GL_SYSDIRSEQ,-      ;SET NEW SYSTEM DIRECTORY SEQ NUM.
              14 A0      0735                 LNMC$L_SYSDIRSEQ(R0)
              18 A0  7C  0737         CLRQ    LNMC$L_ENTRY(R0)          ;CLEAR FIRST ENTRIES
          62     60  0E  073A  50$:   INSQUE  (R0),(R2)                 ;INSERT AT HEAD OF QUEUE
                     05  073D         RSB
                         073E
        50    04 B2  0F  073E  80$:   REMQUE  @LNMC$L_BLINK(R2),R0      ;TAKE OLDEST ENTRY
                 DF  1C  0742         BVC     40$                       ;IF THERE ARE ANY
                 50  D4  0744  100$:  CLRL    R0                        ;NO ENTRY
                     05  0746         RSB
                         0747
                         0747  ;      .PAGE
                     0747            .SBTTL  LNM$PROBER      - PROBE LOGICAL NAME DESCRIPTOR FOR READ ACCESS
                     0747      ;+
                     0747      ; LNM$PROBER - PROBE LOGICAL NAME DESCRIPTOR FOR READ ACCESS
                     0747      ;
                     0747      ; THIS ROUTINE IS CALLED TO PROBE A DESCRIPTOR FOR A LOGICAL NAME FOR READ
                     0747      ; ACCESS.  IF CHECK THE LENGTH OF THE DESCRIPTOR FOR VALIDITY AS WELL
                     0747      ; AS CHECKING ACCESS TO THE DESCRIBED BUFFER.
                     0747      ; ACCESS TO THE DESCRIPTOR IS NOT CHECKED.
                     0747      ;
                     0747      ; INPUTS:
                     0747      ;
                     0747      ;       R0 = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR
                     0747      ;
                     0747      ; OUTPUTS:
                     0747      ;
                     0747      ;       R0 LOW BIT CLEAR INDICATES FAILURE TO TRANSLATE.
                     0747      ;
                     0747      ;               R0 = SS$_ACCVIO - ACCESS VIOLATION.
                     0747      ;               R0 = SS$_IVLOGNAM - INVALID LOGICAL NAME.
                     0747      ;                       R1 AND R2 ARE MODIFIED.
                     0747      ;
                     0747      ;       R0 LOW BIT SET INDICATES SUCCESS.
                     0747      ;
                     0747      ;               R0 = SS$_NORMAL - DESCRIPTOR IS VALID
                     0747      ;                       R1 = LENGTH OF BUFFER IN BYTES.
                     0747      ;                       R2 = ADDRESS OF BUFFER.
                     0747      ;-
                     0747      LNM$PROBER::
      51   60   7D   0747              MOVQ    (R0),R1                 ;FETCH DESCRIPTOR
      51   51   3C   074A              MOVZWL  R1,R1                   ;GET LENGTH OF LOGICAL NAME STRING
           11   13   074D              BEQL    10$                     ;IF EQL INVALID LOGICAL NAME
                     074F              ASSUME  LNM$C_NAMLENGTH LE 512
 00FF 8F   51   B1   074F              CMPW    R1,#LNM$C_NAMLENGTH     ;LEGAL NAME STRING LENGTH?
           0A   1A   0754              BGTRU   10$                     ;IF LEQU YES
                     0756              IFNORD  R1,(R2),20$             ;CAN LOGICAL NAME STRING BE READ?
      50   01   D0   075C              MOVL    #SS$_NORMAL,R0          ;SUCCESS STATUS
                05   075F              RSB
 50  0154  8F   3C   0760      10$:    MOVZWL  #SS$_IVLOGNAM,R0        ;SET INVALID LOGICAL NAME
                05   0765              RSB
      50   0C   3C   0766      20$:    MOVZWL  #SS$_ACCVIO,R0          ;SET ACCESS VIOLATION
                05   0769              RSB
                     076A
                     076A      ;       .PAGE
```

```
                        076A          .SBTTL  LNM$LOCKR       - LOCK LOGICAL NAME TABLE FOR READ ACCESS
                        076A          .SBTTL  LNM$LOCKW       - LOCK LOGICAL NAME TABLE FOR WRITE ACCESS
                        076A    ;+
                        076A    ; LNM$LOCKR - LOCK LOGICAL NAME TABLE FOR READ ACCESS
                        076A    ; LNM$LOCKW - LOCK LOGICAL NAME TABLE FOR WRITE ACCESS
                        076A    ;
                        076A    ; THESE ROUTINES ARE CALLED TO SYNCHRONIZE ACCESS TO LOGICAL NAME TABLES.
                        076A    ;
                        076A    ; INPUTS:
                        076A    ;
                        076A    ;       R4 = CURRENT PROCESS PCB ADDRESS.
                        076A    ;
                        076A    ; OUTPUTS:
                        076A    ;
                        076A    ;       REGISTER R0 IS MODIFIED.
                        076A    ;       REGISTERS R1, R2, AND R3 ARE PRESERVED ACROSS CALL.
                        076A    ;-
                        076A
                        076A          .ENABL  LSB
                        076A    LNM$LOCKR::                             ;LOCK LOGICAL NAME TABLE FOR READ ACCESS
        00000000'EF  9F 076A          PUSHAB  L^SCH$LOCKR             ;SET ADDRESS OF LOCK ROUTINE
                 0E  11 0770          BRB     10$                     ;
                        0772    LNM$LOCKW::                             ;LOCK LOGICAL NAME TABLE FOR WRITE ACCESS
        00000000'EF  9F 0772          PUSHAB  L^SCH$LOCKW             ;SET ADDRESS OF LOCK ROUTINE
                 06  11 0778          BRB     10$                     ;
                        077A    ;       .PAGE
```

```
                                   077A  2644          .SBTTL  LNM$UNLOCK       - UNLOCK LOGICAL NAME TABLE
                                   077A  2645 ;+
                                   077A  2646 ; LNM$UNLOCK - UNLOCK NAME TABLE
                                   077A  2647 ;
                                   077A  2648 ; THIS ROUTINE IS CALLED TO UNLOCK LOGICAL NAME TABLES AND ALLOW ACCESS BY
                                   077A  2649 ; OTHER PROCESSES.
                                   077A  2650 ;
                                   077A  2651 ; INPUTS:
                                   077A  2652 ;
                                   077A  2653 ;       R4 = CURRENT PROCESS PCB ADDRESS.
                                   077A  2654 ;
                                   077A  2655 ; OUTPUTS:
                                   077A  2656 ;
                                   077A  2657 ;       R0, R1, R2, AND R3 ARE MODIFIED.
                                   077A  2658 ;
                                   077A  2659 ;-
                                   077A  2660
                                   077A  2661 LNM$UNLOCK::                                  ;UNLOCK NAME TABLE
          00000000'EF       9F     077A  2662          PUSHAB  L^SCH$UNLOCK                 ;SET ADDRESS OF LOCK ROUTINE
     50   00000000'EF       DE     0780  2663 10$:     MOVAL   L^LNM$AL_MUTEX,R0            ;GET ADDRESS OF LOGICAL NAME TABLE MUTEX
                            05     0787  2664          RSB                                  ;PERFORM SYNCHRONIZATION OPERATION
                                   0788  2665          .DSABL  LSB
                                   0788  2666
                                   0788  2667          .END
```

```
                                                          F  2
LNMSUB                        - LOGICAL NAME RELATED SUBROUTINES      16-SEP-1984 00:30:35  VAX/VMS Macro V04-00        Page  59
Symbol table                                                           5-SEP-1984 03:44:03  [SYS.SRC]LNMSUB.MAR;1           (31)

ARM$M_READ                  = 00000001                   LNMB$T_NAME                 = 00000011
ARM$M_WRITE                 = 00000002                   LNMB$V_NODELETE             = 00000004
CAS_MEASURE                 = 00000002                   LNMB$V_NO_ALIAS             = 00000000
CHPCTL$B_MODE               = 00000008                   LNMB$V_TABLE                = 00000003
CHPCTL$C_LENGTH             = 0000000C                   LNMB$W_SIZE                 = 00000008
CHPCTL$L_ACCESS             = 00000000                   LNMC$B_CACHEINDX            = 0000000B
CHPCTL$L_FLAGS              = 00000004                   LNMC$K_NUM_ENTRIES          = 0000001A
CHPCTL$M_READ               = 00000001                   LNMC$L_BLINK                = 00000004
CHPCTL$M_USEREADALL         = 00000004                   LNMC$L_ENTRY                = 00000018
CHPCTL$M_WRITE              = 00000002                   LNMC$L_FLINK                = 00000000
CTL$GL_LNMDIRSEQ              ********    X   02         LNMC$L_PROCDIRSEQ           = 00000010
CTL$GQ_LNMTBLCACHE            ********    X   02         LNMC$L_SYSDIRSEQ            = 00000014
DELETE_ENTRY                  00000000 R      02         LNMC$L_TBLADDR              = 0000000C
DELETE_LNMB                   0000004F R      02         LNMHSH$K_BUCKET             = 0000000C
DELETE_NAMES                  000000C4 R      02         LNMHSH$L_MASK               = 00000000
DELETE_TABLE                  00000101 R      02         LNMTH$B_FLAGS               = 00000000
DYN$C_ORB                   = 00000049                   LNMTH$L_BYTES               = 00000021
EXE$CHKPRO_INT                ********    X   02         LNMTH$L_BYTESLM             = 0000001D
EXE$CLEANUP_ORB               ********    X   02         LNMTH$L_CHILD               = 00000011
EXE$DEAP1                     ********    X   02         LNMTH$L_HASH                = 00000001
EXE$DEAPAGED                  ********    X   02         LNMTH$L_NAME                = 00000009
EXE$UPCASE_DAT                ********    X   02         LNMTH$L_ORB                 = 00000005
LNM$AL_DIRTBL                 ********    X   02         LNMTH$L_PARENT              = 0000000D
LNM$AL_HASHTBL                ********    X   02         LNMTH$L_QTABLE              = 00000019
LNM$AL_MUTEX                  ********    X   02         LNMTH$L_SIBLING             = 00000015
LNM$CHECK_PROT                00000131 RG     02         LNMTH$V_DIRECTORY           = 00000001
LNM$CONTSEARCH                000004F5 RG     02         LNMTH$V_GROUP               = 00000002
LNM$C_MAXDEPTH              = 0000000A                   LNMTH$V_SYSTEM              = 00000003
LNM$C_NAMLENGTH             = 000000FF                   LNMX$B_FLAGS                = 00000000
LNM$DELBLK                    00000192 RG     02         LNMX$B_INDEX                = 00000001
LNM$DELETE                    000001A6 R      02         LNMX$C_BACKPTR              = FFFFFF81
LNM$DELETE_HASH               000001FF RG     02         LNMX$C_TABLE                = FFFFFF82
LNM$DELETE_LNMB               000001BF RG     02         LNMX$T_XLATION              = 00000004
LNM$DELETE_TAB                0000021A RG     02         LNMX$V_TERMINAL             = 00000001
LNM$FIRSTTAB                  000004AF RG     02         LNMX$V_XEND                 = 00000002
LNM$GL_SYSDIRSEQ              ********    X   02         LNMX$W_HASH                 = 00000002
LNM$HASH                      0000056E RG     02         LRSB                          000005EF R      02
LNM$INIT_PROT                 0000022B RG     02         NT_B_ACMODE                 = 00000000
LNM$INSLOGTAB                 0000026F RG     02         NT_B_FLAGS                  = 00000001
LNM$LOCKR                     0000076A RG     02         NT_K_LENGTH                 = 00000014
LNM$LOCKW                     00000772 RG     02         NT_L_NAMADR                 = 00000008
LNM$LOOKUP                    000005B2 R      02         NT_L_NAMLEN                 = 00000004
LNM$PRESEARCH                 000004CB RG     02         NT_L_TABID                  = 0000000C
LNM$PROBER                    00000747 RG     02         NT_L_THREAD                 = 00000010
LNM$SEARCHLOG                 000003A5 RG     02         NT_M_CASE                   = 00000001
LNM$SEARCH_ONE                00000438 RG     02         NT_M_MODIFY                 = 00000004
LNM$SETUP                     000005F0 RG     02         NT_V_CASE                   = 00000008
LNM$TABLE                     00000616 RG     02         NT_V_MODIFY                 = 0000000A
LNM$TABLE_SRCH                00000669 R      02         NT_W_HASH                   = 00000002
LNM$TBL_CACHE                 000006F0 R      02         NT_W_RS                     = 00000000
LNM$UNLOCK                    0000077A RG     02         ORB$B_FLAGS                 = 0000000B
LNM$V_CREATE_IF             = 00000018                   ORB$B_TYPE                  = 0000000A
LNMB$B_ACMODE               = 0000000B                   ORB$C_LENGTH                = 00000058
LNMB$B_FLAGS                = 00000010                   ORB$K_LENGTH                = 00000058
LNMB$L_BLINK                = 00000004                   ORB$L_ACL_COUNT             = 00000028
LNMB$L_FLINK                = 00000000                   ORB$L_ACL_DESC              = 0000002C
LNMB$L_TABLE                = 0000000C                   ORB$L_ACL_MUTEX             = 00000004
```

```
                                                           G  2
LNMSUB                          - LOGICAL NAME RELATED SUBROUTINES      16-SEP-1984 00:30:35  VAX/VMS Macro V04-00        Page  60
Symbol table                                                             5-SEP-1984 03:44:03  [SYS.SRC]LNMSUB.MAR;1             (31)

ORB$L_GRP_PROT                  = 00000020
ORB$L_OWNER                     = 00000000
ORB$L_OWN_PROT                  = 0000001C
ORB$L_SYS_PROT                  = 00000018
ORB$L_WOR_PROT                  = 00000024
ORB$Q_MODE_PROT                 = 00000010
ORB$R_MAX_CLASS                 = 00000044
ORB$R_MIN_CLASS                 = 00000030
ORB$S_MAX_CLASS                 = 00000014
ORB$S_MIN_CLASS                 = 00000014
ORB$W_REFCOUNT                  = 0000000E
ORB$W_SIZE                      = 00000008
PCB$L_ARB                       = 0000008C
PCB$L_UIC                       = 000000BC
PCB$Q_PRIV                      = 00000084
PMS$GL_LOGNAM                     ********    X    02
PR$_IPL                           ********    X    02
PRV$V_GRPNAM                    = 00000003
PRV$V_SYSNAM                    = 00000002
PSL$C_USER                      = 00000003
RT_B_ACMODE                     = 00000000
RT_B_DEPTH                      = 00000002
RT_B_FLAGS                      = 00000001
RT_B_TRIES                      = 00000003
RT_C_MAXTRIES                   = 000000FF
RT_K_LENGTH                     = 00000030
RT_L_CACHEPTR                   = 00000004
RT_L_STACK                      = 00000008
RT_M_CASE                       = 00000001
RT_M_TERM                       = 00000002
RT_V_CASE                       = 00000008
RT_V_TERM                       = 00000009
RT_W_R5                         = 00000000
SCH$LOCKR                         ********    X    02
SCH$LOCKW                         ********    X    02
SCH$UNLOCK                        ********    X    02
SS$_ACCVIO                      = 0000000C
SS$_DUPLNAM                     = 00000094
SS$_IVLOGNAM                    = 00000154
SS$_IVLOGTAB                    = 0000015C
SS$_LNMCREATED                  = 000006B1
SS$_NOLOGNAM                    = 000001BC
SS$_NOLOGTAB                    = 00002294
SS$_NOPRIV                      = 00000024
SS$_NORMAL                      = 00000001
SS$_PARENT_DEL                  = 00002254
SS$_SUPERSEDE                   = 00000631
SS$_TOOMANYLNAM                 = 00000374

                                                 +----------------+
                                                 ! Psect synopsis !
                                                 +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (   0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)  01 (   1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
```

```
                                                         H  2
LNMSUB                         - LOGICAL NAME RELATED SUBROUTINES       16-SEP-1984 00:30:35  VAX/VMS Macro V04-00        Page  61
Psect synopsis                                                           5-SEP-1984 03:44:03  [SYS.SRC]LNMSUB.MAR;1            (31)

YF$$LNM                        00000788  ( 1928.)  02 (  2.)  NOPIC    USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE

                                          +------------------------+
                                          ! Performance indicators !
                                          +------------------------+

Phase                       Page faults    CPU Time        Elapsed Time
-----                       -----------    --------        ------------
Initialization                      35     00:00:00.05     00:00:01.39
Command processing                 119     00:00:00.53     00:00:04.44
Pass 1                             380     00:00:13.48     00:00:44.17
Symbol table sort                    0     00:00:01.66     00:00:05.10
Pass 2                             399     00:00:05.69     00:00:17.73
Symbol table output                  1     00:00:00.13     00:00:00.70
Psect synopsis output                0     00:00:00.02     00:00:00.02
Cross-reference output               0     00:00:00.00     00:00:00.00
Assembler run totals               936     00:00:21.56     00:01:13.55

The working set limit was 2100 pages.
85200 bytes (167 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1052 non-local and 102 local symbols.
2667 source lines were read in Pass 1, producing 18 object records in Pass 2.
23 pages of virtual memory were used to define 22 macros.

                                         +----------------------------+
                                         ! Macro library statistics !
                                         +----------------------------+

Macro library name                               Macros defined
------------------                               --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                             10
_$255$DUA28:[SYSLIB]STARLET.MLB;2                           9
TOTALS (all libraries)                                     19

1115 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:LNMSUB/OBJ=OBJ$:LNMSUB MSRC$:LNMSUB/UPDATE=(ENH$:LNMSUB)+EXECML$/LIB
```

0376 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

IOSUBRAMS
LIS

IPCONTROL
LIS

IOSUBPAGD
LIS

LNMSUB
LIS

IOPERFORM
LIS

IOSUBNPAG
LIS

LINKVEC
LIS

0377 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY